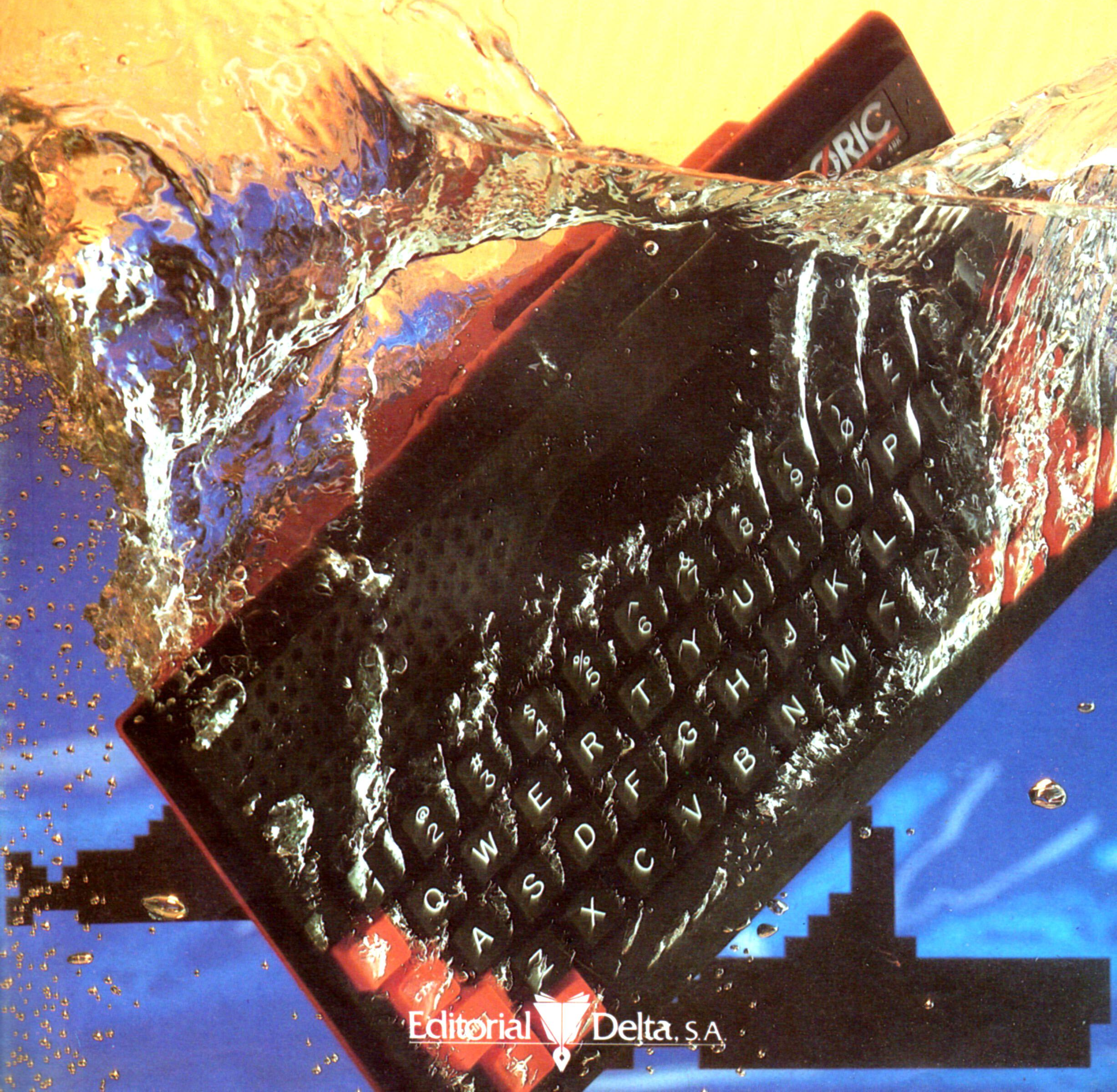
150 ptas.

mi computer

38

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 38

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

© 1983 Orbis Publishing Ltd., London
© 1984 Editorial Delta, S.A., Barcelona
ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 038410
Impreso en España - Printed in Spain - Septiembre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.



Devorando caracteres

Un procesador de textos permite que la tarea de producir cartas, informes o ensayos resulte de una gran sencillez para el usuario

En realidad los procesadores de textos no son otra cosa que versiones informatizadas de la máquina de escribir. El texto se digita a través del teclado del ordenador y aparece en la pantalla. Se pueden realizar modificaciones fácilmente sin necesidad de volver a teclear el documento entero y, una vez que la redacción es correcta, el texto se imprime por una impresora de ordenador.

Aparte del temeroso respeto que existe hacia los ordenadores, lo único que probablemente disuade a la gente de hacer uso de un procesador de textos es el problema de no sentirse cómodo al utilizar un teclado. Y, sin embargo, es más fácil iniciarse empleando un teclado con un procesador de textos que con una máquina de escribir normal. Los errores inevitables del mecanógrafo improvisado que al utilizar por primera vez una máquina de escribir usa sólo dos dedos pueden dar lugar a un lío considerable. Con un procesador de textos, estos errores se pueden enmendar en cuestión de segundos.

Para el tratamiento de textos se puede utilizar prácticamente cualquier micro personal, pero algunos no son tan prácticos como otros. En algunos casos ello se debe a que no existe un buen software para tratamiento de textos para esa máquina en particular; en otros, es el propio micro y sus periféricos los que no son adecuados para la tarea. Incluso el precio de un sistema simple para tratamiento de textos puede ser bastante elevado, ya que los accesorios necesarios pueden fácilmente costar el doble de lo que cuesta el propio ordenador. Por lo general el elemento más caro suele ser la impresora. Sin disponer de una buena impresora, de poco vale poseer un paquete para tratamiento de textos. En el futuro inmediato, casi todo el texto procesado ha de terminar imprimiéndose en papel; la edad del correo electrónico, en la que todo el texto se enviará directamente de un micro a otro, aún está muy lejana.

Hasta las impresoras más sencillas son bastante caras y, aun así, la calidad de la impresión que producen es relativamente baja. En muchos casos el tratamiento de textos exige una impresión de gran calidad. Al fin y al cabo, no tiene mucho sentido pasarse el tiempo con un procesador de textos tratando de que el aspecto de una carta sea perfecto, si el resultado se va a imprimir luego en una impresora matricial. Las impresoras margarita ofrecen una superior calidad de impresión, pero son lentas y caras, si bien los precios están descendiendo con bastante rapidez. Algunas máquinas de escribir electrónicas se pueden dotar de interfaces de modo que se las pueda utilizar como impresoras para ordenador.

Una solución al problema de tener acceso a una impresora de calidad es que varios amigos o un club informático compartan el costo de la máquina entre todos. Los usuarios aún tendrían la dificultad de conectar sus micros con interfaces para poder utilizar la impresora. Con algunos ordenadores esto es fácil porque poseen interfaces estándar, de modo que cada usuario sólo ha de adquirir un cable apropiado para conectar su micro a la impresora. Algunos micros, como los Commodore o los Atari, tienen interfaces que los limitan a las impresoras de su propia marca. Unos pocos micros, entre los cuales



BBC Micro
La combinación del BBC Micro y el cartucho de disco Torch permite utilizar software de gestión, incluyendo el excelente Wordstar. El costo del sistema, sin embargo, es más elevado que el de muchos micros de oficina. Usando una impresora más económica y, en lugar del Wordstar, el software Perfect Writer ("el escritor perfecto"), incluido en el precio del cartucho Torch, se reduciría el precio de manera sustancial

Ian McKinnell

está incluido el Sinclair Spectrum, no tienen interface para impresora como tal, por lo que se debe comprar la interface como accesorio adicional.

Disponiendo de una impresora adecuada, su siguiente tarea consiste en elegir el software apropiado para tratamiento de textos. Para las marcas más populares se produce una amplia gama de programas, mientras que las máquinas menos populares disponen sólo de uno o dos. La calidad varía considerablemente entre los distintos programas. Algunos son limitados y sólo permiten una edición sencilla, como insertar y eliminar textos. Otros permiten desplazar párrafos enteros de un lugar a otro del texto, presentarlo en la pantalla tal como aparecerá sobre el papel o justificar sus márgenes (igualar el largo de las líneas, espaciando convenientemente las palabras que las integran, como sucede en el texto que usted está leyendo en estos momentos).

Algunos procesadores de textos pueden buscar una palabra o una frase determinadas, de modo que resulta fácil corregir un error de ortografía que se hubiera deslizado a lo largo de un artículo. Para ciertos procesadores de textos se venden programas que verifican la ortografía de cada una de las palabras empleadas en un texto; y otros programas, como una lista de direcciones o una base de datos, pueden trabajar conjuntamente con el procesador de textos. Los programas para tratamiento de textos más sofisticados están diseñados para aprovechar las posibilidades ofrecidas por ciertas impresoras. Con frecuencia las impresoras matriciales pueden producir varios tipos diferentes de letras (como cursiva, negrita o pequeña) y, por tanto, algunos procesadores de textos permiten mezclar en el mismo artículo distintos tipos. Algunos procesadores de textos, aunque pocos, se pueden ampliar para utilizar la capacidad de producir gráficos de ciertas impresoras matriciales.

Las impresoras margarita pueden utilizar *espaciado proporcional*, dándoles más espacio a las letras anchas como la "w" y menos a las estrechas, como la "i", en vez de concederles a todas el mismo espacio, como lo haría una máquina de escribir convencional. Algunos procesadores de textos pueden utilizar esta facilidad, que hace que el texto resulte mucho más legible, y así y todo arreglárselas para justificar ambos márgenes. Esto es ideal para editar periódicos comunitarios o revistas internas de clubes, porque proporciona un aspecto profesional sin incurrir en el gasto de la correcta composición tipográfica. Las ruedas de impresión intercambiables permiten elegir diversos tipos de letras para elaborar el artículo.

El software para tratamiento de textos se vende en diversos formatos, incluyendo cinta, disco, cartucho y chip de ROM. No obstante, aún más importante es la forma en que se almacena el texto procesado, por lo general en cinta o en disco flexible. Aunque económica, la cinta es engorrosa, lenta y limita la longitud de los artículos a las dimensiones de la memoria disponible. Los discos son mejores porque son rápidos, fiables y permiten escribir artículos extensos. Actualmente están comenzando a aparecer nuevas formas de almacenamiento de datos. El microdrive de Sinclair, por ejemplo, a pesar de ser barato puede almacenar grandes cantidades de datos y hallar en cuestión de segundos una parte determinada del texto. Sin embargo, por ahora, son pocos los procesadores de textos para el Spectrum capaces de trabajar con los microdrives. Otro sistema interesante es la unidad de cinta que utiliza el Coleco Adam, un micro personal evidentemente diseñado teniendo en mente el tratamiento de textos, puesto que incluye una impresora margarita. Utiliza cintas de cassette modificadas para almacenar sus datos y en éstas se puede localizar cualquier palabra en segundos.

Guardar una copia procesada por tratamiento de textos en cinta o en disco permite escribir artículos largos en el transcurso de varios días, utilizar cartas estándar muchas veces, y conservar copias de todo el trabajo. También es una buena idea, al escribir artículos largos, hacer copias en diversas etapas mientras se los va escribiendo. Si esto no se hace existe el peligro de que algún accidente, como un corte de fluido eléctrico, destruya todo el trabajo realizado.

Algunos programas poseen instrucciones extrañas y combinaciones de teclas difíciles de memorizar, mientras que otros son de fácil uso. Algunos



Sinclair Spectrum
Éste es un sistema económico, y a la vez eficaz, para tratamiento de textos, pero aun así es demasiado caro. El Spectrum impone varias limitaciones, incluyendo un teclado pobre, la falta de una interface para pantalla y el uso de microdrives en vez de unidades de disco. No obstante, es posible añadir un teclado de mejor calidad. Tasword Two es uno de los pocos procesadores de textos para el Spectrum que funcionan con los microdrives

Ian McKinnell



micros, como el Sinclair Spectrum, poseen teclados de poca calidad que no contribuyen a facilitar la labor del usuario. Por suerte, para el Spectrum existen teclados accesorios que lo colocan en un estándar aceptable. Los que cuentan con teclas de función extras son útiles porque reducen el número de códigos de funciones que es necesario memorizar. La visualización en pantalla también puede ser una fuente de problemas. Varios micros muestran en la pantalla muy poco texto cada vez, lo que dificulta la escritura de manera notable. El Commodore Vic-20, por ejemplo, sólo visualiza texto en una anchura de 22 caracteres, mientras que las máquinas de oficina suelen tener una anchura de pantalla de 80 caracteres. El micro ideal para una fuerte carga de trabajo de proceso de textos es aquel que proporciona al menos una visualización de 25 por 80 caracteres y que utiliza una pantalla adecuada para visualizar clara y nítidamente la imagen.

El diseño de las letras que se ven en la pantalla del ordenador varía considerablemente. Algunas máquinas construyen cada letra con más puntos que otras máquinas, lo que hace que la visualización resulte más fácil de leer y también que sea más cómodo trabajar con ella. Son pocos los micros personales que utilizan letras de sólo seis puntos de anchura; la mayoría emplea una matriz de ocho puntos de anchura. Unas pocas máquinas de oficina poseen caracteres de 16 por 16 puntos, que son de una calidad superlativa.

Los sistemas personales más modestos son aptos para escribir cartas y otros textos cortos, pero se necesita más equipo para que resulte práctico escribir libros o informes largos. Si ha de procesar gran cantidad de texto, necesita un sistema con una pantalla, dos unidades de disco, una buena impresora, un teclado del tipo máquina de escribir y un buen software para tratamiento de textos. Ampliar un micro personal para que alcance este nivel resulta caro; de hecho, suele resultar más oneroso que adquirir un verdadero micro de oficina.

Los ordenadores de oficina ofrecen otras ventajas. Al estar diseñados para satisfacer necesidades de gestión, poseen buenos teclados, pantallas, unidades de disco o interfaces para impresora. Con todo, su ventaja más importante radica en la gran calidad del software creado para ellos. Tal gama de software de calidad existe porque la mayoría de los ordenadores de oficina utiliza alguno de los escasos sistemas operativos estandarizados, lo que significa que cada programa para tratamiento de textos se vende para muchas máquinas diferentes.

Con mucho, el paquete de gestión para tratamiento de textos más conocido en Gran Bretaña es el Wordstar, disponible para sistemas operativos CP/M, CP/M-86 y MS-DOS. El programa tiene algunas características sofisticadas, pero es caro, ya que cuesta alrededor de 20 veces más que un programa medio para micros personales. El costo de un procesador de textos de gestión es elevado, pero utilizar un micro personal para gran cantidad de proceso de textos por parte de una empresa pequeña o incluso de un eficiente escritor, es una falsa economía. El tiempo que se pierde empleando un sistema limitado superará la cantidad ahorrada.

El costo de los sistemas de gestión serios es elevado, pero está disminuyendo constantemente. Algunos micros personales se pueden ampliar para que utilicen sistemas operativos estándares tales como el CP/M, de modo que las personas que ya han invertido mucho dinero en un sistema de micro personal están empezando a disponer de software de calidad sin pagar demasiados extras. Otra tendencia está contribuyendo a reducir el costo del tratamiento de textos. Varias empresas están incluyendo en sus ordenadores, y sin recargo adicional alguno, programas de tratamiento de textos como el Wordstar.

La novedad más reciente en cuanto a tratamiento de textos es el tratamiento de textos sobre la marcha. Varios ordenadores de pilas se venden con procesadores de textos incorporados para permitir que los ajetreados ejecutivos escriban recordatorios y cartas en cualquier lugar y en cualquier momento. Estas máquinas no están al alcance del presupuesto de la mayoría de los usuarios de micros personales y no poseen muchos otros usos. Pero las máquinas de mano (*hand-held*) parecen sugerir que el tratamiento de textos se está convirtiendo en algo cada vez más común. Tal vez no sea sólo la máquina de escribir la que está condenada a desaparecer, sino también el lápiz y el papel.



Commodore 64
El Commodore 64 ofrece un procesador de textos con una unidad de disco de precio muy razonable. Tener una sola unidad de disco constituye una limitación, y el Commodore 64 sólo produce una visualización de 40 caracteres de ancho

Ian McKinnell

Diseño de sprites

La creación de sprites es una de las características más atractivas de los gráficos del Commodore 64

Un sprite es una gran forma gráfica móvil. Se diseña de modo muy similar a los caracteres definidos por el usuario de ocho por ocho que ya hemos analizado anteriormente en el curso (véase p. 713), pero se construye en un cuadriculado mucho más grande. Una vez definido un sprite, características tales como el color y la posición en pantalla se controlan mediante un juego de registros especiales en el chip de control de video (o VIC) del Commodore 64.

Un sprite se compone de 21 filas de 24 pixels. Cada fila consta de tres segmentos de ocho pixels y se representa mediante tres bytes de memoria, de modo que se requieren 63 bytes en total para almacenar los datos de un sprite. Al igual que sucede con los caracteres definidos por el usuario, cada pixel del cuadriculado del sprite que se iluminará en su forma final se representa mediante un uno binario (y los pixels no iluminados mediante un cero binario). Por consiguiente, para cada fila del sprite podemos calcular los equivalentes decimales de cada grupo de ocho dígitos binarios. Los diagramas que proporcionamos aquí muestran los cuatro sprites que se utilizarán en el juego *Subhunter*. Los números que figuran al lado de cada esquema son los equivalentes decimales que formarán las sentencias de datos para cada sprite (tal como se especifican desde la línea 6000 a la 6370 del programa).

Una vez el sprite ha sido definido y convertido en una serie de sentencias DATA, los datos se deben leer (READ) y colocar (POKE) en la memoria. Los datos de los sprites se pueden situar en varias posiciones de memoria. Por ejemplo, utilizando las posiciones que comienzan en 12288 el sprite se colocará en la zona para programas en BASIC, que va de 2048 a 40960. A medida que se va entrando un programa BASIC en el Commodore 64, éste va ocupando espacio de memoria desde la posición 2048 en adelante. Un programa tendría que tener un tamaño de 10 Kbytes antes de que alcanzara la posición 12288 y, por tanto, machacara los datos del sprite. No obstante, cuando un programa se está ejecutando, todas las variables utilizadas se almacenan en la zona superior a la empleada para almacenar el programa (las variables alfanuméricas o en serie, en particular, se van guardando hacia abajo desde el tope de la zona para programas BASIC). Como el juego *Subhunter* utiliza la variable de temporizado, TI$, la actualización regular de su valor y su subsiguiente almacenamiento pueden machacar la zona en que deseamos almacenar los datos del sprite.

Una solución a este problema consiste en bajar el tope de la zona para programas BASIC colocándolo por debajo de la zona en la que se guardan los datos del sprite. El puntero de la dirección del tope de memoria se guarda en las posiciones 55 (byte bajo) y 56 (byte alto). Normalmente estas dos posiciones contienen los valores 0 y 160 respectivamente, que representan la dirección 40960. En la forma bajo-alto, la posición 12288 se consigna como 0 y 48. Podemos bajar el tope de la memoria a esta posición simplemente colocando (POKE) estos valores en las posiciones 55 y 56 al comienzo del programa (véase línea 90).

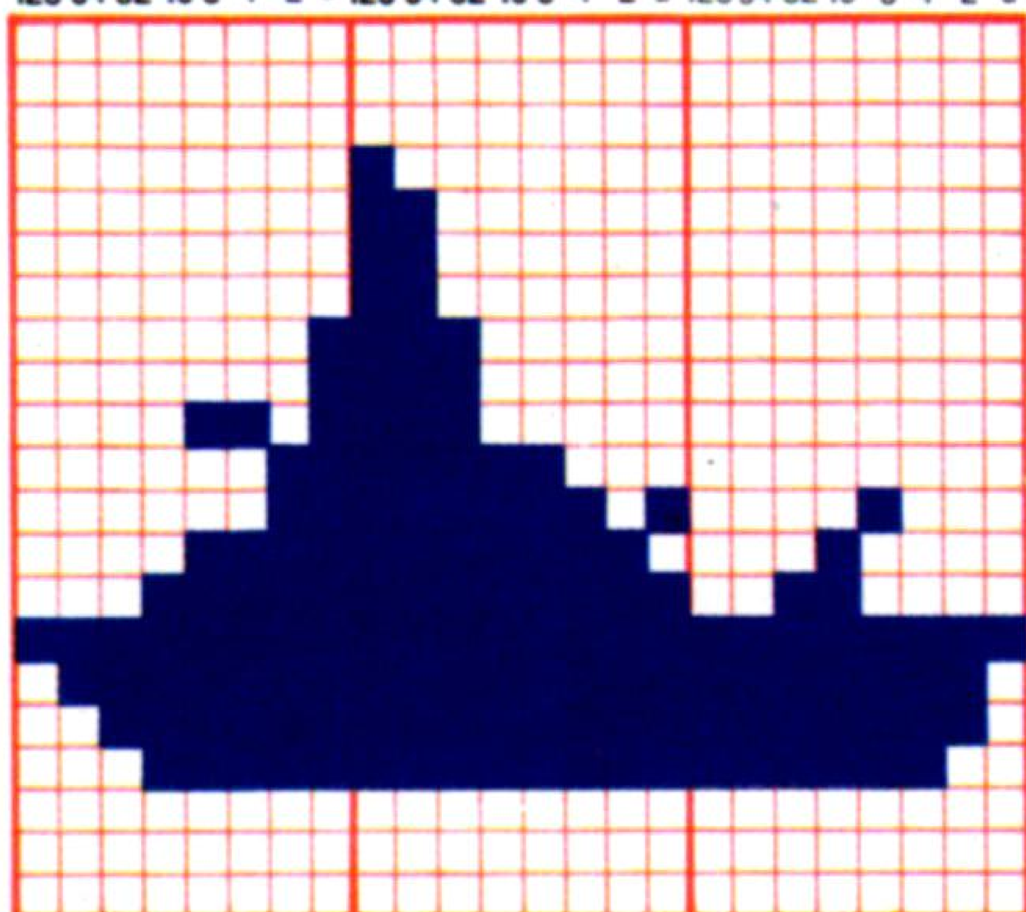
Punteros de sprites

Dado que los datos del sprite se pueden situar en diversas partes de la memoria, se necesita un puntero que indique dónde empiezan los datos. Hay ocho punteros de sprites, guardados entre las posiciones 2040 (para el sprite 0) y 2047 (para el sprite 7). El valor guardado en cada puntero de sprites alude a la zona que contiene los datos del sprite mediante esta fórmula: comienzo de los 63 bytes de datos = (puntero de sprites) x 64. Los datos para el barco de nuestro programa empiezan en 12288 y se designará al barco como el sprite 0, de modo que el puntero de la posición 2040 es 192 (12288/64). El siguiente bloque de datos es para la explosión, el

Blanco móvil
Un sprite se compone de 21 filas de tres bytes; estos bytes en realidad son patrones de bits, y en las líneas de datos del programa BASIC se almacenan como sus equivalentes en números decimales. Estos valores se ven junto a los diagramas de los sprites y en el listado del programa. El programa coloca (POKE) los valores en una zona exclusiva de RAM, donde son accedidos por el chip controlador de video como datos de sprite, visualizándolos y moviéndolos por la pantalla con un mínimo esfuerzo de programación

BARCO-SPRITE 0

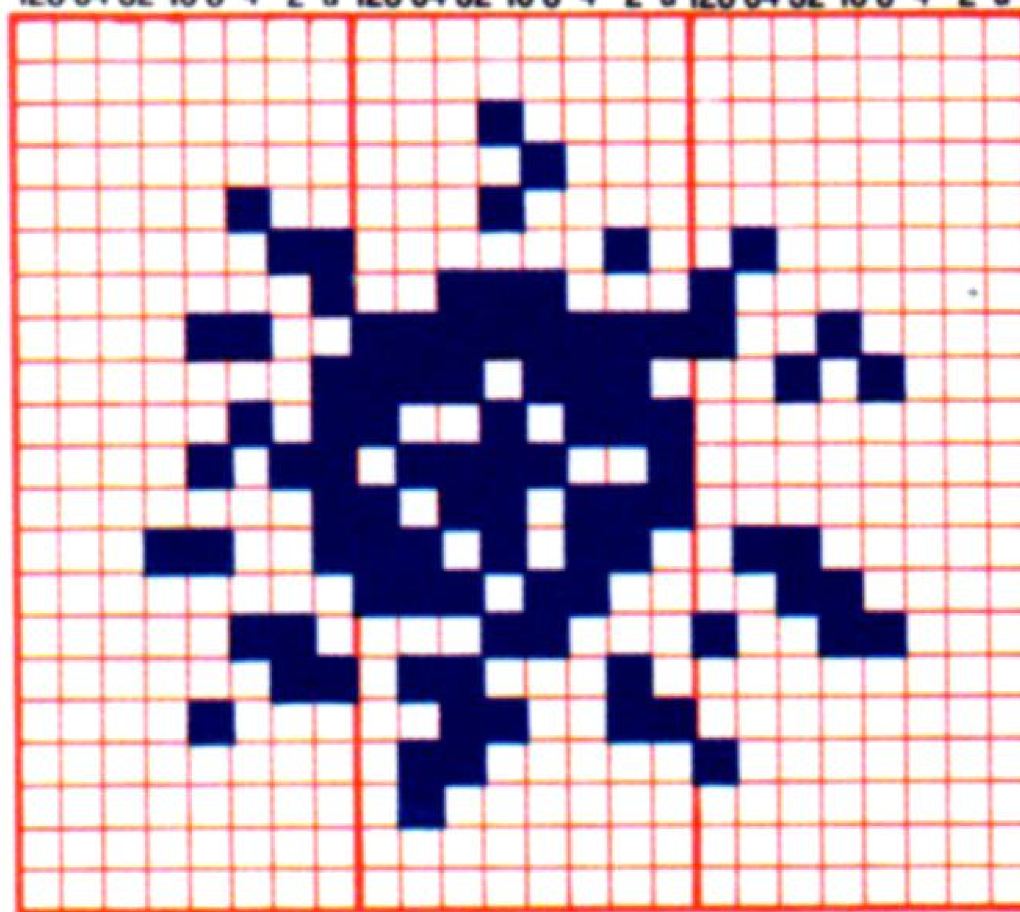
1 2 3
128 64 32 16 8 4 2 u 128 64 32 16 8 4 2 u 128 64 32 16 8 4 2 u



1	2	3
0	0	0
0	0	0
0	0	0
0	128	0
0	192	0
0	192	0
0	192	0
1	224	0
1	224	0
13	224	0
3	248	128
3	253	8
15	254	16
31	255	48
255	255	255
127	255	254
63	255	254
31	255	252
0	0	0
0	0	0
0	0	0

EXPLOSIÓN-SPRITE 1

1 2 3
128 64 32 16 8 4 2 u 128 64 32 16 8 4 2 u 128 64 32 16 8 4 2 u



1	2	3
0	0	0
0	0	0
0	16	0
0	8	0
4	16	0
3	2	64
1	56	128
12	255	144
1	238	40
5	151	0
11	121	0
1	183	0
25	214	96
0	236	48
6	24	152
3	98	0
8	51	0
0	96	128
0	64	0
0	0	0
0	0	0



sprite 1. Si ponemos el puntero de la posición 2041 en 193, entonces los datos deben comenzar en 12352. Los valores que utilizamos son éstos:

Número de sprites	0	1	2	3
Puntero de sprites	192	193	194	195
63 bytes de datos del sprite	de 12288 a 12350	de 12352 a 12414	de 12416 a 12478	de 12480 a 12542

Observe que al final de cada bloque de datos de sprite queda un byte sin utilizar. Las partes del programa que leen los datos del sprite de la memoria y especifican los punteros de sprites están contenidos en las líneas 2000 a 2210.

Manipulación de sprites

El chip de control de video (VIC: *VIdeo Control chip*) tiene varios registros especiales que se utilizan para controlar los sprites. La primera posición del chip VIC es la 53248 y para nuestro programa es más fácil describir las posiciones de todos los demás registros en relación a ésta. Si hacemos que V=53248, la siguiente posición del chip VIC, la 53249, se puede denominar V+1, y así sucesivamente. V se debe definir, junto con otras variables, en una etapa anterior (véase línea 100).

El color de cada uno de los sprites se establece colocando (POKE) un número de código de color (comprendido entre 0 y 15) en un registro especial. Cada uno de los ocho sprites posee su propio registro de color; éstos van de V+39 a V+46. Por ejemplo, para colorear de negro el barco simplemente colocamos (POKE) el código de color 0 en la posición V+39. Los otros sprites se pueden colorear de la misma manera (véase desde línea 2220 a 2250).

El posicionamiento de los sprites en la pantalla lo analizaremos con mayor profundidad en el próximo capítulo. Por ahora basta con saber que la coordenada *x* del sprite 0 se guarda en la posición V, la coordenada *y* para el sprite 0 se guarda en la posición V+1; las coordenadas *x* e *y* para el sprite 1 se guardan en V+2 y V+3 respectivamente, y así sucesivamente hasta la posición V+15 (véase desde línea 2260 a 2280).

Los sprites se pueden ampliar horizontalmente, verticalmente o en ambas direcciones en un factor de dos. Los sprites del barco y del submarino podrían parecen más bien aplastados horizontalmente, pero ahora los ampliaremos al doble de su longitud original. De hecho, los cuatro sprites se ampliarán horizontalmente. El registro del chip VIC que controla la expansión horizontal es V+29, que es más fácil de utilizar que los otros registros que hemos visto. En vez de emplear ocho registros diferentes para controlar los atributos de cada uno de los ocho sprites, todo lo que hay que hacer es activar o desactivar la función. Por lo tanto, sólo se requiere un bit del registro para controlar la ampliación horizontal de cada sprite. Para ampliar horizontalmente un sprite, hay que poner a 1 el bit correspondiente del registro V+29. La siguiente tabla muestra el POKE que se requiere para ampliar los cuatro sprites que hemos definido:

Número de sprite	7	6	5	4	3	2	1	0	
Contenido de V+29	0	0	0	0	1	1	1	1	=15 (decimal)

La ampliación en dirección vertical se controla mediante V+23. La explosión, el sprite 1, se amplía vertical y horizontalmente, doblando, por tanto, su tamaño (véase desde línea 2290 a 2310).

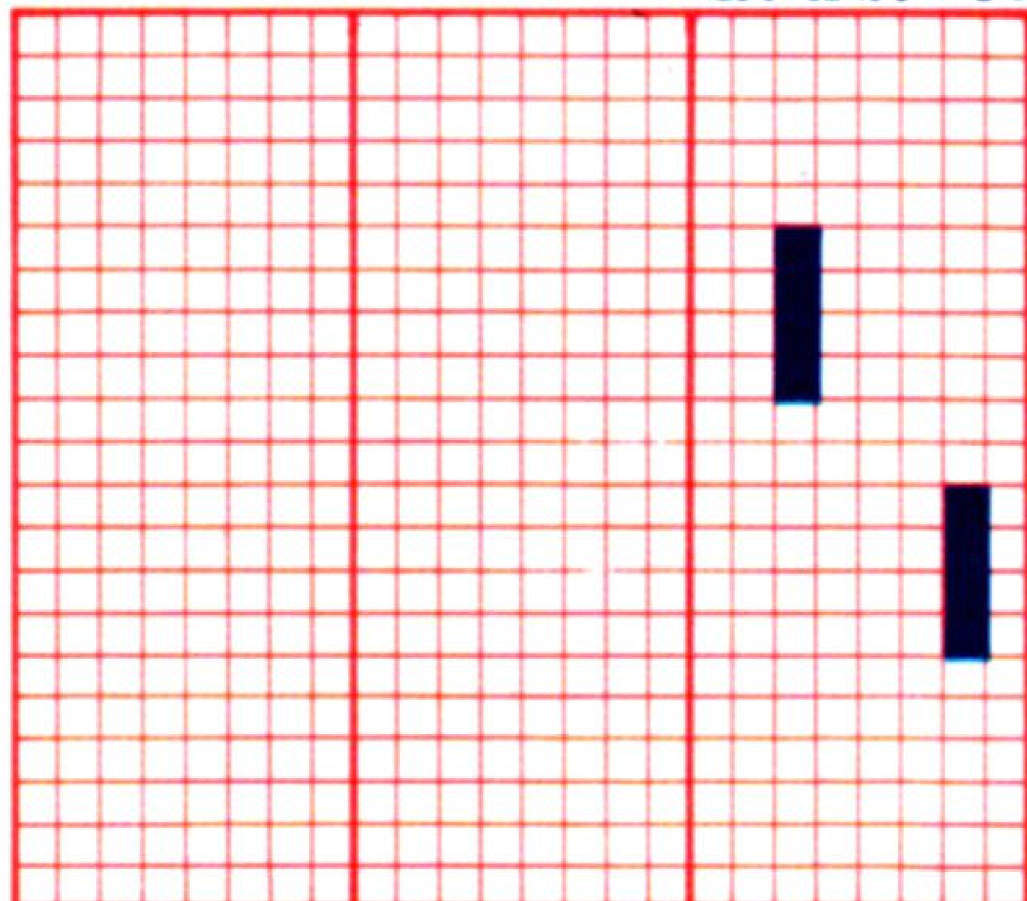
Nuestra tarea final consiste en activar los sprites requeridos. Para activar o desactivar cada sprite se utiliza un solo bit del registro, V+21. En el juego *Subhunter* inicialmente sólo se encienden el barco y el submarino (líneas 2310 a 2360).

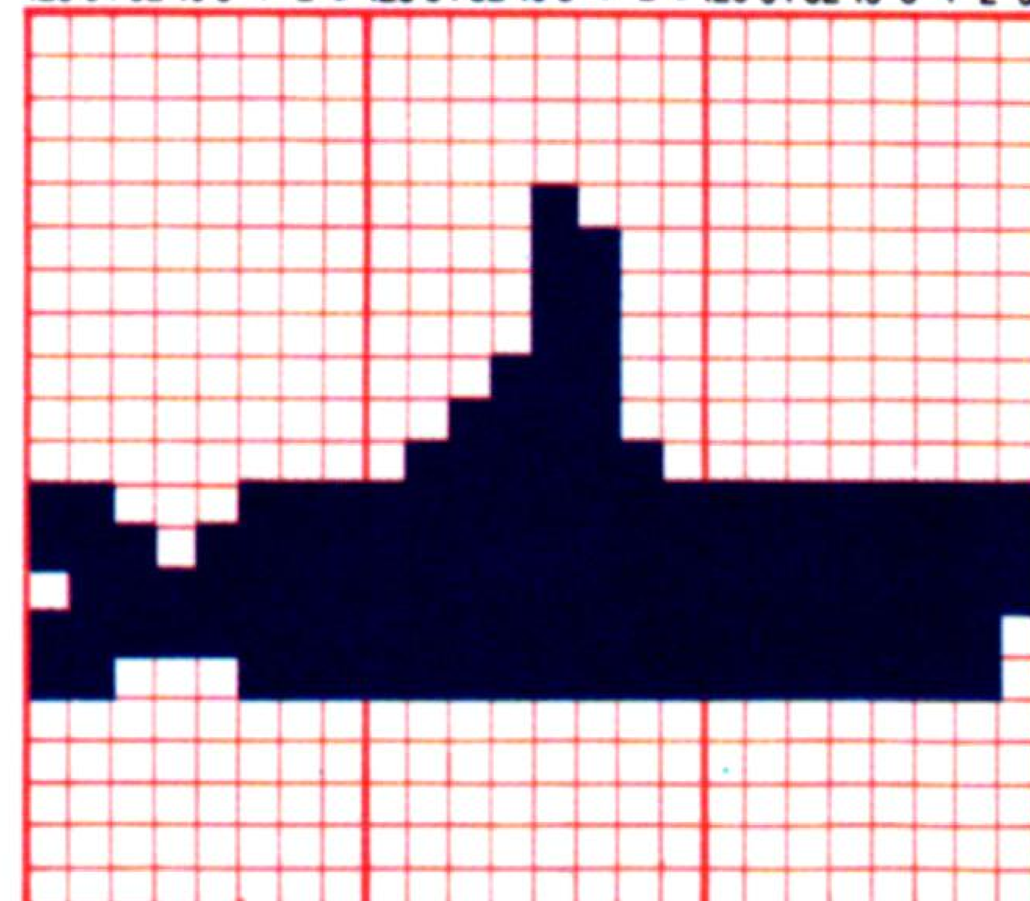
Después de entrar todo el programa, deberá comprobar que los datos de los sprites se hayan leído correctamente. Para hacerlo, ejecute el programa e interrúmpalo mediante RUN o STOP cuando en la pantalla aparezca el reloj. Entrando las siguientes sentencias, sin los números de línea, se posicionarán y visualizarán los cuatro sprites.

```
POKEV,160               (coordenada del barco)
POKEV+2,240:            (coordenadas x e y de
   POKEV+3,100          la explosión)
POKEV+4,160:            (coordenadas x e y de
   POKEV+5,100          la carga de profundidad)
POKEV+6,100:            (coordenadas x e y
   POKEV+7,100          del submarino)
POKEV+21,15             (enciende sprites 0–3)
```

Si el programa se interrumpe con un mensaje "OUT OF DATA ERROR", verifique cuántos números hay en las sentencias DATA. Debería haber 63 para cada sprite. Si el programa se cuelga y el teclado no responde, asegúrese de que en la línea 100 se haya declarado V. Siempre es una buena idea guardar (SAVE) su programa antes de ejecutarlo.

```
1 REM *** GRAFICOS C64 ***********
90  POKE 55,0:POKE 56,48:CLR: REM
    BAJAR TOPE MEM
100 V=53248:FL=0:SC=0
110 GOSUB 1000 :REM CREACION
    PANTALLA (véase p. 695)
120 GOSUB 2000 :REM CREACION
    SPRITES
2000 REM *** CREACION SPRITES ***
2020 REM ** LEER DATOS BARCO **
2030 FOR I=12288 TO 12350
2040 READ A:POKE I,A:NEXT I
2060 REM ** LEER DATOS EXPLOSION **
2070 FOR I=12352 TO 12414
2080 READ A:POKE I,A:NEXT I
2100 REM ** LEER DATOS CARGA **
2110 FOR I=12416 TO 12478
2120 READ A:POKE I,A:NEXT I
2140 REM ** LEER DATOS SUBM **
2150 FOR I=12480 TO 12542
2160 READ A:POKE I,A:NEXT I
2180 REM ** ESTABLECER PUNTEROS **
2190 POKE 2040,192:POKE 2041,
     193:POKE 2042,194:POKE 2043,195
2220 REM ** ESTABLECER COLORES **
2230 POKE V+39,0:POKE V+40,1:POKE
     V+41,0:POKE V+42,0
2260 REM ** INIC COORD. BARCO ***
2270 POKE V+1,80:X0=160
2290 REM ** EXPANDIR SPRITES ***
2300 POKE V+29,15:POKE V+23,2
2320 REM ** ENCENDER SPRITES ***
2330 POKE V+21,9
2340 RETURN
2350 :
6000 REM ** DATOS BARCO ***
6010 DATA 0,0,0,0,0,0,0,0,0
6020 DATA 0,128,0,0,192,0,0,192,0
6030 DATA 0,192,0,1,224,0,1,224,0
6040 DATA 13,224,0,3,248,128,3,253,8
6050 DATA 15,254,16,31,255,48,255,
     255,255
6060 DATA 127,225, 254,63,255,254,
     31,255,252
6070 DATA 0,0,0,0,0,0,0,0,0
6100 REM ** DATOS EXPLOSION ***
6110 DATA 0,0,0,0,0,0,16,0,0,8,0,4,16
6120 DATA 0,3,2,64,1,56,128,12,255,144
6130 DATA 1,238,40,5,151,0,11,121,0,1
6140 DATA 183,0,25,214,96,0,236,48,
     6,24
6150 DATA 152,3,98,0,8,51,0,0,96,128,0
6160 DATA 64,0,0,0,0,0,0,0
6200 REM ** DATOS CARGA PROF ***
6210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6220 DATA 0,0,0,32,0,0,32,0,0,32,0,
     0,32,0
6230 DATA 0,0,0,0,0,0,0
6240 DATA 2,0,0,2,0,0,2,0,0,2,0,0
6250 DATA 0,0,0,0,0,0,0,0,0
6260 DATA 0,0,0,0,0,0,0,0,0
6300 REM ** DATOS SUBMARINO ***
6310 DATA 0,0,0,0,0,0,0,0,0,0,0,0
6320 DATA 0,8,0,0,12,0,0,12,0
6330 DATA 0,12,0,0,28,0,0,60,0
6340 DATA 0,126,0,199,255,255
6350 DATA 239,255,255,127,255,255
6360 DATA 255, 255, 254, 199,255,254
6370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```





Liz Dixon

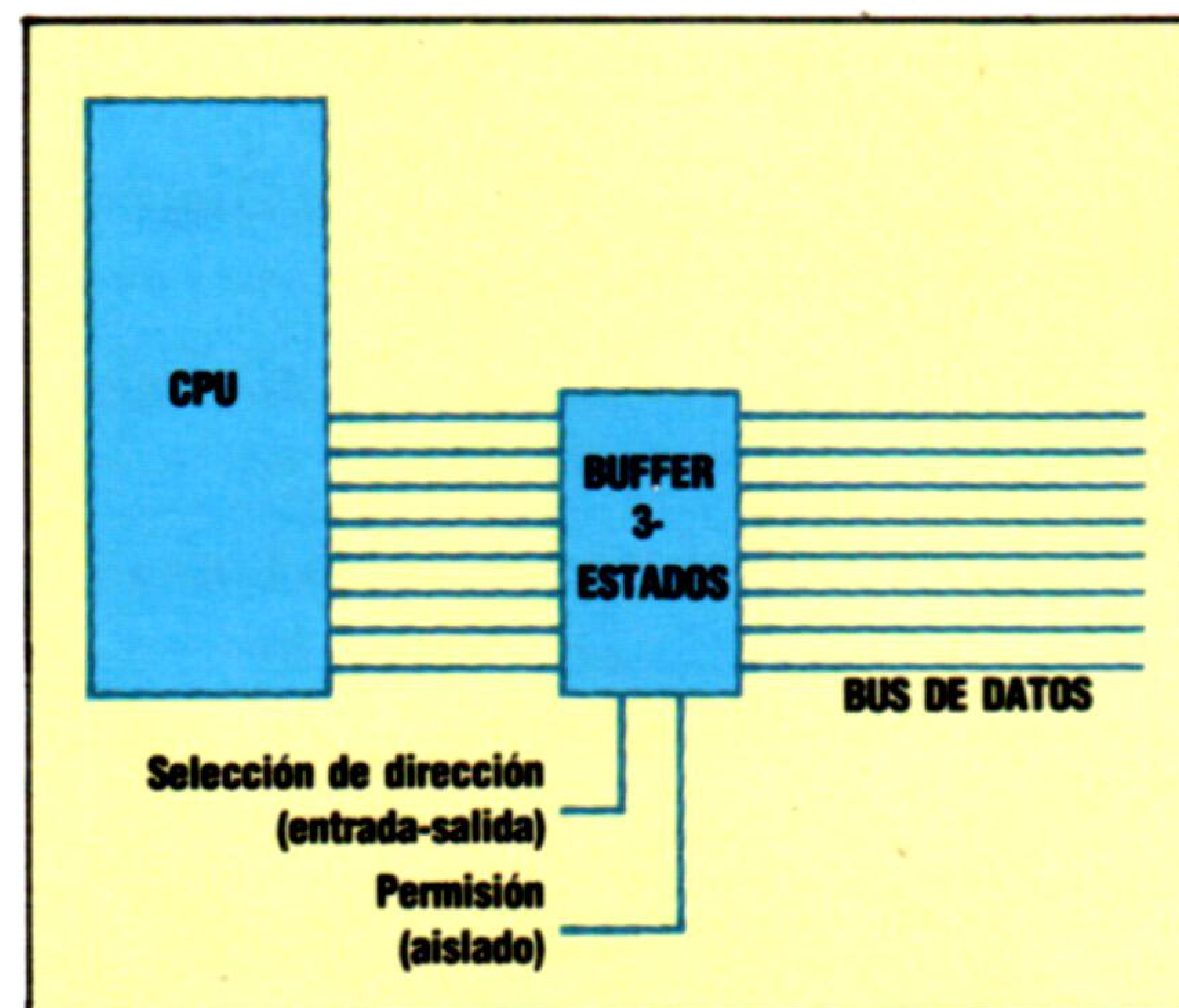
De aquí para allá

En este capítulo veremos cómo se produce el intercambio de datos entre la CPU y la memoria

Cada posición de memoria de un ordenador personal normalmente se compone de ocho bits. Los datos se pueden transferir, de a ocho bit por vez, mediante una serie de ocho líneas paralelas, a la CPU, donde los datos se pueden entonces utilizar de acuerdo a la instrucción de programa que los requiere. Los datos también se pueden enviar en la dirección contraria, para almacenarlos en una posición de memoria. La instrucción en código máquina LDA $1234 hace que el número contenido en la posición $1234 se envíe hasta la CPU por el bus de datos. STA $1234 hace que se envíe un número desde la CPU, nuevamente por el bus de datos, y que se lo almacene en la posición $1234.

Por consiguiente, el bus de datos debe permitir transferencias en ambas direcciones. En ciertas ocasiones también es importante aislar la CPU del bus de datos. Por lo tanto, cada línea del bus de datos puede estar en uno de los tres estados (INPUT —entrada—, OUTPUT —salida— o ISOLATE —aislado—). A fin de conseguir la necesaria conmutación entre estos estados, cada línea del bus de datos posee un pequeño circuito electrónico llamado *dispositivo 3-estados (tri-state)*.

Ocho de esos dispositivos 3-estados se combinan formando un único circuito integrado. El diagrama siguiente muestra cómo este circuito integrado enlaza al bus de datos con la CPU. Asimismo, el diagrama ilustra las líneas "permisión" (*enable*) y "selección de dirección", que ponen los ocho 3-estados en el estado operativo requerido. Los circuitos de este tipo también se pueden utilizar para conectar otros dispositivos, como periféricos de entrada/salida, con el bus de datos.



Siempre que deseamos cargar el contenido de una posición determinada nos referimos a la posición que ocupa mediante su dirección. Cada posición de ROM y RAM posee su propio número exclusivo que hace referencia a ella. Veamos ahora, a nivel de hardware, cómo se puede acceder a cada posición para realizar una transferencia de datos.

La mayoría de los microordenadores posee una segunda vía entre la CPU y la memoria, que se denomina *bus de direcciones*. Normalmente el bus de direcciones no posee ocho líneas sino 16. Esto significa que se pueden especificar hasta 65 536 direcciones distintas (2^{16}=65 536). Es decir, utilizando un bus de direcciones de 16 bits se puede acceder a 64 Kbytes de memoria. Se puede pensar en el total de memoria como si estuviera dividida en módulos, conteniendo cada uno de ellos 256 posiciones. Los ocho bits inferiores de la dirección se pueden entonces utilizar para hallar la posición concreta dentro de un módulo dado. El módulo propiamente dicho se puede seleccionar utilizando algunos (o la totalidad) de los ocho bits de dirección restantes.

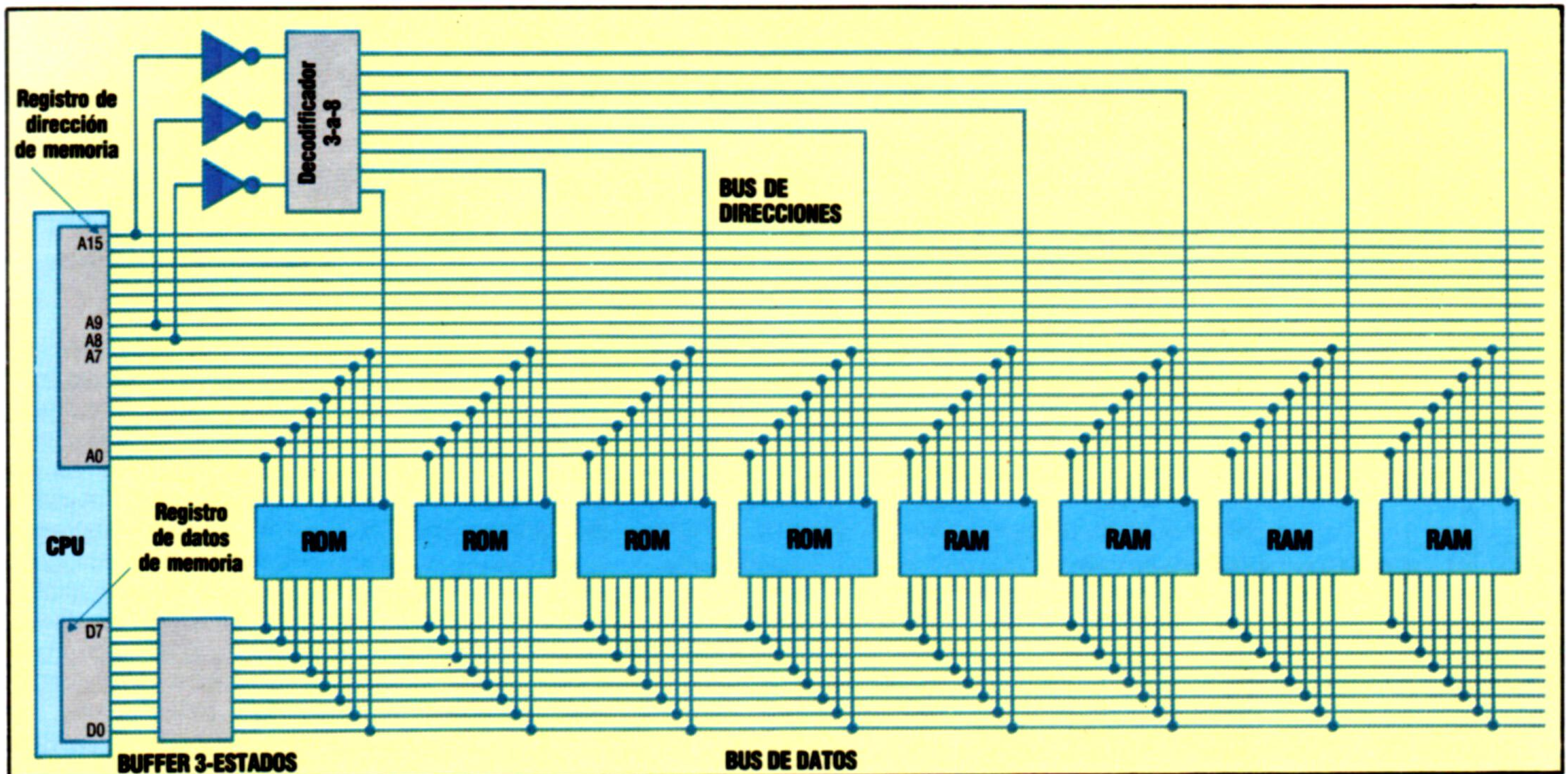
Si consideramos el sencillo ejemplo de un microordenador con un tamaño total de memoria de dos Kbytes, podemos ver cómo tiene lugar la selección de cualquier posición determinada. Dado que cada módulo de memoria contiene 256 posiciones, nuestro ordenador de dos Kbytes requerirá ocho módulos. Para nuestro pequeño ordenador daremos por sentado que la memoria está dividida en ROM y RAM a partes iguales.

La dirección de la posición requerida se guarda en un registro de 16 bits especial de la CPU, llamado registro de dirección de memoria o MAR (*Memory Address Register*). Como los ocho bits inferiores de la dirección seleccionan una posición particular dentro de cualquier módulo, las ocho líneas inferiores del bus de direcciones se pueden conectar a cada uno de los módulos de la memoria. Para seleccionar un módulo determinado, ya sólo necesitamos otros tres bits (2^3=8). Este código de tres bits se debe decodificar en ocho líneas de salida.

El diagrama muestra cómo los módulos de memoria se enlazan con la CPU a través de estos buses de datos y direcciones. Cada módulo de memoria tiene una única línea hacia él desde el decodificador de tres(bits)-a-ocho (líneas). Tres de los bits de dirección superiores se utilizan para determinar qué módulo se ha de seleccionar. Si se hubieran de agregar más módulos de RAM, entonces para seleccionar cualquier módulo individual, se podrían usar algunos de los cinco bits superiores restantes.

Después de haber visto cómo se puede seleccionar una posición determinada de memoria y transferir los datos, analicemos cómo la CPU lleva a cabo una instrucción de lenguaje máquina. Todo programa en lenguaje máquina se suele almacenar en posiciones consecutivas. Una instrucción puede requerir dos o tres bytes para su almacenamiento. Una instrucción como ADD $13FF significa "sumarle al acumulador el contenido de la posición cuya dirección hexadecimal es $13FF". Esta instrucción requeriría tres bytes: uno para guardar el código bi-



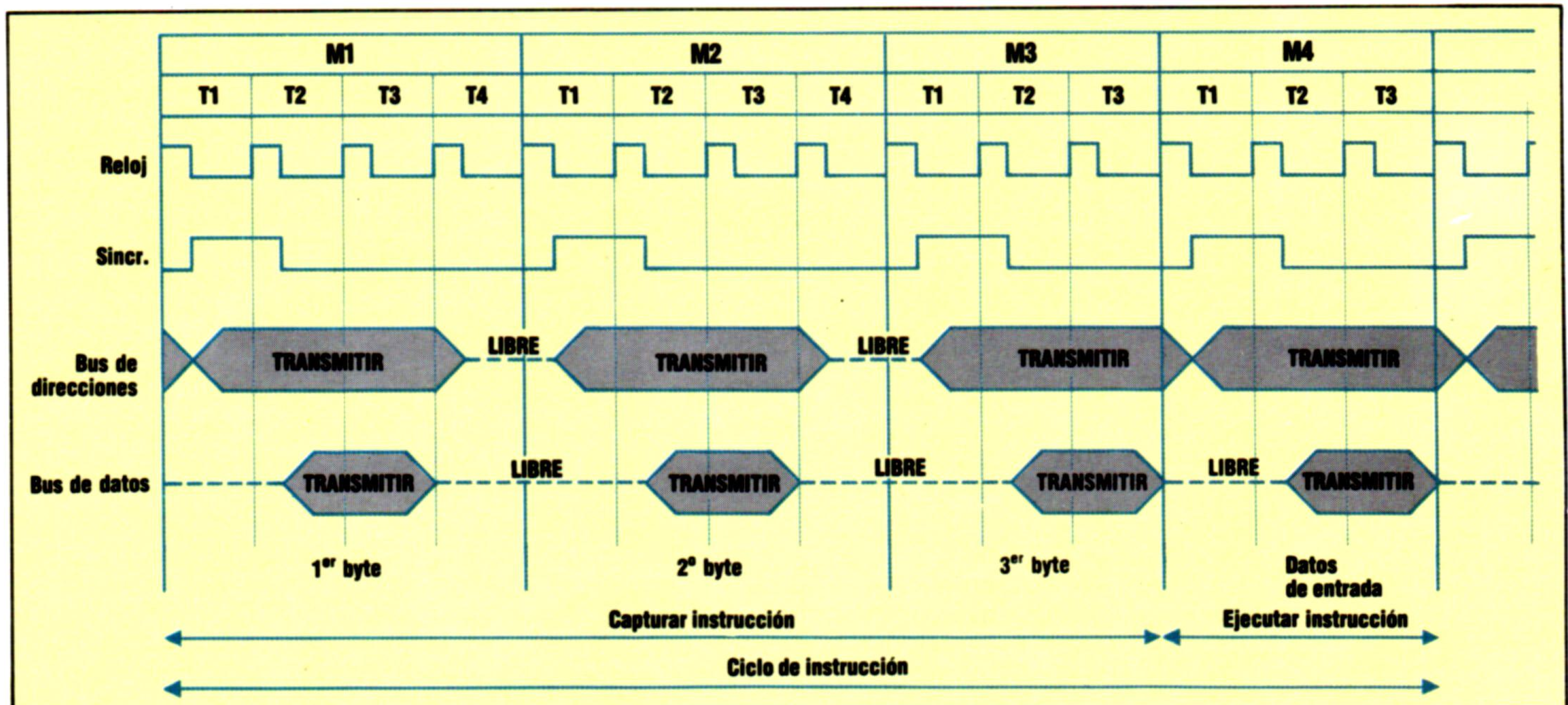


nario de la instrucción ADD y dos para contener la dirección de 16 bits, $13FF. Vamos a suponer que se almacena en las posiciones $1000, $1001 y $1002.

Antes de que la instrucción se pueda procesar, debe ser "capturada" (operación *fetch*) de la memoria. Ello requiere tres accesos separados para llevar los tres bytes a lo largo del bus de datos hasta la CPU. Al completar el ciclo de captura, la instrucción entera está en un registro especial dentro de la CPU. Lo único que hay que hacer entonces es decodificar la instrucción y obedecerla. La instrucción del ejemplo requiere otro acceso a memoria para obtener el contenido de la posición $13FF de modo que pueda ser sumado al acumulador.

Todos los fabricantes de ordenadores publican las características de sus procesadores en forma de diagramas de tiempos. Éstos muestran el orden de los acontecimientos para numerosas operaciones diferentes del ordenador. Podemos trazar un diagrama de tiempos para los ciclos de captura y ejecución de una instrucción de lenguaje máquina. El temporizado de las operaciones se controla mediante los impulsos del reloj (véase p. 726) y nuestro gráfico muestra que en este sistema imaginario el bus de direcciones es activado por el flanco de cabeza del impulso de sincronización, mientras que el bus de datos es activado por el flanco de cola de sincronización. El impulso de sincronización, por su parte, es activado por el flanco de cola del primer impulso de reloj de cualquier fase operativa, o ciclo de máquina. Los ciclos tienen duraciones diferentes porque el procesador necesita más tiempo para decodificar el byte de código de instrucción que para manipular los bytes de los operandos: el código se debe decodificar de inmediato porque especifica el número de bytes de los operandos.

Capturar y ejecutar
Una instrucción en lenguaje máquina que conste de un byte de código de instrucción seguido de dos bytes de operandos se trata en un ciclo de instrucción que consta de fases de captura y de ejecución. Durante la fase de captura, el bus de direcciones accede a las posiciones de memoria que contienen la instrucción, y el bus de datos lleva los bytes de instrucción hasta la CPU. Allí, los buses de datos y de direcciones todavía están atareados durante la fase de ejecución, porque la instrucción que se está ejecutando provoca un acceso a la memoria

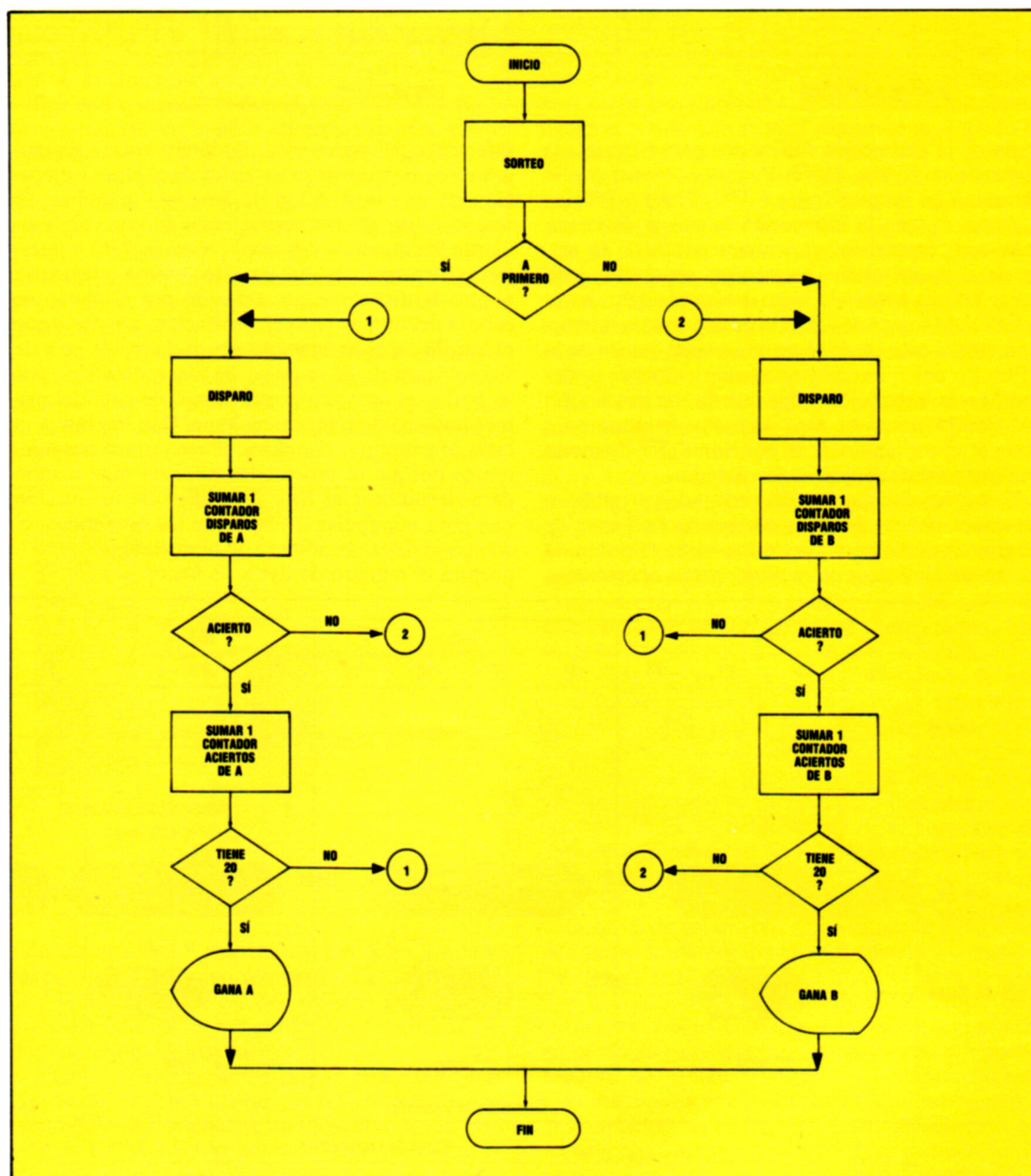


Uso de contadores

Un juego de tiro al blanco se puede prestar muy bien a ser controlado por programa

Dos jugadores, A y B, practican el tiro al plato, y deciden que ganará el primero que consiga veinte aciertos. Se sortea quién comienza el juego. Empieza a disparar el jugador seleccionado: A. Éste va tirando hasta que falla un plato; en ese momento cede el turno de disparos al jugador B, que dispara hasta fallar, y así sucesivamente. El juego termina cuando uno de los jugadores llega a contabilizar 20 en su cuenta de aciertos. Entonces se visualizará, por una parte, cuál de los jugadores que participaban en el juego ha ganado y, por otra, cuántos disparos le han hecho falta para conseguir la victoria.

En este caso serán necesarios cuatro contadores, dos por jugador: uno contará el número de disparos efectuados y el otro el número de aciertos. En esta ocasión se prescindirá de cualquier intervención manual, todo será controlado por programa. A este fin, el sorteo del jugador que empieza a disparar, y si el disparo es o no fallido, se determinará según un número aleatorio.





De la misma estirpe

Después del discreto éxito obtenido por el Oric-1 en su lanzamiento en 1983, Oric Products ha creado un nuevo y mejorado modelo

Equipado con un potente BASIC estilo Microsoft, una puerta para impresora Centronics incorporada y un conector para pantalla RGB estándar, el Oric-1 parecía originalmente una buena inversión. Sin embargo, la falta de un buen software, junto con algunos notorios errores en la ROM de BASIC, hicieron que la máquina fuera recibida con indiferencia.

Ahora Oric Products International ha corregido los errores fundamentales y ha relanzado la máquina con el nombre de Oric Atmos. El antiguo teclado tipo calculadora ha sido sustituido por teclas más profesionales, tipo máquina de escribir y de recorrido total, y se ha rediseñado la carcasa con una elegante combinación en rojo y negro. La disposición del teclado es la misma que en el Oric-1, con una tecla Function adicional, que aún está desconectada pero que se suministra en virtud de una "futura ampliación".

El Atmos utiliza un microprocesador 6502 y en operación normal tiene 37 Kbytes de RAM libres para programas BASIC. El Atmos puede visualizar ocho colores y posee una resolución máxima de 240 × 200 pixels. El juego de caracteres se mantiene en RAM, lo que permite al usuario la definición de cualquier carácter. También existe un juego de caracteres alternativo, que ofrece gráficos de bloque de tipo teletexto. A diferencia del Spectrum, que mantiene un archivo de atributos independiente en RAM, el Atmos utiliza "atributos en serie". Éstos emplean menos memoria pero se visualizan en la pantalla como espacios en blanco, por lo que hay que tener mucho cuidado al planificar las visualizaciones en pantalla.

La ROM del Atmos contiene cuatro sonidos preestablecidos (ZAP, PING, SHOOT y EXPLODE) y éstos dan efectos de sonido al estilo de los juegos recreativos. Las órdenes MUSIC, PLAY y SOUND le permiten al usuario sacar el máximo provecho del sofisticado chip de sonido del Oric, disponiendo de una amplia gama de parámetros para hacer variar el sonido. El volumen oscila desde muy débil a muy fuerte, y los tres canales de tonos y un canal de ruido proporcionan una escala de siete octavas.

El BASIC del Oric original incluyó varios molestos *bugs*. La orden TAB no funcionaba correctamente y la visualización a menudo se alteraba por las órdenes de sonido. El Oric también introducía códigos de control erróneos al evaluar la función STR$, y



Ian McKinnell

El sistema Atmos
El Oric Atmos es un ordenador personal de moderado precio con 48 K de memoria, gráficos en color y efectos de sonido. Oric fabrica dos accesorios para el Atmos, ambos en colores que hacen juego con el conjunto. La unidad de disco proporciona una alternativa rápida en contraposición a la grabadora de cassette, y la impresora-plotter puede trazar líneas o texto en color



El Oric-1
El Atmos es una versión mejorada del Oric-1. Utiliza la misma placa de circuito impreso pero tiene un chip de ROM diferente que contiene una versión perfeccionada de BASIC. Estas modificaciones son suficientes para hacer del Atmos una máquina mucho mejor. Es necesario advertir a los usuarios que gran parte del software de Oric no funcionará en el Atmos, por lo que deberán reconvertir sus programas favoritos

daba resultados incorrectos al usar LEN o VAL. La nueva ROM ha solventado estas dificultades. Un inevitable efecto secundario de estas mejoras es el hecho de que es poco probable que los programas del Oric en lenguaje máquina funcionen con el Atmos, porque varias rutinas de ROM han sido reubicadas en memoria.

El BASIC es una versión ampliada del dialecto Microsoft, desarrollada por Tansoft a partir del BASIC Tangerine original. Admite la estructura IF... THEN...ELSE completa (en esta instrucción el BASIC del Oric-1 tenía un error en el segmento ELSE) y también la instrucción de bucle REPEAT...UNTIL. Una característica inusual es el disponer de las instrucciones POP y PULL, que se utilizan para saltar fuera de rutinas GOSUB y REPEAT...UNTIL sin provocar un mensaje de error. El BASIC del Oric-1 no permitía que el usuario utilizara POKE con un valor hexadecimal; en la nueva ROM esto también se ha corregido.

Las primeras versiones del Atmos tuvieron algunos problemas con la ROM nueva. Al diseñar el nuevo chip, Oric incluyó una rutina para verificación de errores mejorada para cargar las cintas de cassette. Esta rutina era tan eficaz que los usuarios enseguida descubrieron que el software encontraba errores en programas de casi todas las máquinas de cassette. Sin embargo, la ROM diseñada del Oric permite cargar programas de manera satisfactoria.

Coincidiendo con el nuevo Atmos, Oric ha vuelto a diseñar su impresora-plotter, que ahora tiene un acabado en rojo y negro igual que el ordenador. Cuatro pequeños lápices de punta esférica (negro, rojo, verde y azul) están dispuestos en una cabeza giratoria para trazado de gráficos; todos los colores se pueden seleccionar bajo control de software. La impresora-plotter tiene una velocidad lenta para textos, de 12 caracteres por segundo, pero imprime sobre papel normal.

La unidad de microdisco, esperada durante tanto tiempo, también ha sido rediseñada con los colores del Atmos. Oric ha optado por los discos Hitachi de 3"; éstos están metidos dentro de una carcasa rígida de plástico. El Atmos puede utilizar hasta cuatro unidades: una unidad maestra individual con un sistema de interface de disco incorporado y hasta tres unidades esclavas. Por el momento no se

Modulador RF
Convierte la señal de video en señal apta para un televisor normal

Conexión RGB
Permite conectar el Atmos con una pantalla

Conexión TV
Conecta el Atmos a una entrad
de antena estándar de TV

Altavoz
Es muy grande y permite una mejor definición de los sonidos generados

RAM
El Atmos contiene 64 K de RAM, aunque en realidad sólo se utilizan 48 K

Chip de sonido
Este chip le permite al Atmos producir su amplia gama de efectos de sonido





Tecla Function
Esta tecla no está conectada

Interface para impresora
Interface en paralelo tipo Centronics

Puerta de ampliación
Contiene un bus en paralelo de 34 canales para conectar a la unidad de disco

Disipador
Esta placa disipa el calor excesivo que se genera

Versión mejorada de BASIC
El chip individual de ROM contiene la nueva versión del BASIC Tansoft

CPU
La unidad central de proceso es un microprocesador 6502A

Chris Stevens



Unidad de disco
La unidad de disco Oric utiliza discos de 3 pulgadas. Éstos vienen en una carcasa rígida de protección. La unidad posee una capacidad de 160 K en cada cara del disco. Se puede dar la vuelta a los discos y utilizarlos por la otra cara para dar un total de 320 K por disco. Las unidades Oric sólo pueden tratar archivos secuenciales

han producido unidades esclavas, pero se espera que éstas aparezcan pronto. La unidad de disco viene con un transformador de potencia separado lo suficientemente potente para dos unidades de disco y el propio ordenador. En las primeras versiones del sistema operativo en disco surgían problemas cuando se conectaban simultáneamente la impresora y la unidad de disco. Todo intento de editar una línea de programa hacía que ésta se borrara del listado; lo mismo sucedía con todas las líneas de programa numeradas. Oric afirma que las posteriores versiones del sistema operativo han superado esta dificultad.

A pesar de que las primeras versiones han experimentado problemas a la hora de utilizar impresora y cassette, parece ser que Oric Products ha procedido con extremo celo al producir el Atmos y sus periféricos. El equipo de diseño ha tomado nota de las críticas de que fue objeto al anterior Oric-1 y ahora ha enmendado la mayoría de los errores. Los usuarios del Oric-1 estaban muy desatendidos por los productores de software y, para remediar esta situación, Oric Products le ha encargado a Tansoft la producción de un conjunto de programas para utilizar con la unidad de disco. Si la producción de software se incrementa, el Atmos, con toda seguridad, captará un sector más amplio de un mercado que es muy competitivo.



Chris Stevens

Impresora-plotter
Constituye un excelente accesorio para el Atmos. Utiliza cuatro lápices de punta esférica para trazar líneas y texto en color. Puede dibujar textos de tamaño minúsculo hasta varias pulgadas de altura. Entre sus inconvenientes se incluyen la anchura del papel, de sólo 4½ pulgadas, la reducida velocidad y el elevado precio de los lápices de punta esférica. La impresora-plotter no es adecuada para dibujar superficies sólidas de color

ORIC ATMOS

DIMENSIONES
278×178×50 mm

CPU
6502

MEMORIA
48 Kbytes de RAM, 16 Kbytes de ROM

PANTALLA
26 líneas de 40 columnas en modalidad de textos y hasta 200×240 pixels en ocho colores

INTERFACES
Una puerta de ampliación para una interface para impresora Centronics, puerta para cassette y conexión RGB

LENGUAJES DISPONIBLES
BASIC ampliado y FORTH

TECLADO
58 teclas tipo máquina de escribir. La tecla Function no está conectada

DOCUMENTACION
El manual es exhaustivo y está escrito en un agradable estilo coloquial, con la clara intención de que el principiante se inicie fácilmente en la programación en BASIC. Para el usuario más avanzado hay capítulos que cubren programación en lenguaje máquina y técnicas avanzadas de entrada-salida, así como algunos apéndices que proporcionan una completa información técnica

VENTAJAS
El Atmos tiene una amplia gama de facilidades de las que no disponen máquinas de precio más elevado. El BASIC es conciso y contiene numerosas instrucciones que hacen que la programación resulte más sencilla

DESVENTAJAS
El método de visualización en pantalla resulta difícil cuando se trabaja en modalidad de alta resolución. Las unidades de disco son decepcionantes, ya que sólo son aptas para el acceso secuencial

Posiciones clave

En este capítulo estudiaremos dos métodos de acceso a los ficheros aleatorios: el acceso indexado y el acceso mediante hash

Si ha experimentado con los archivos de acceso aleatorio, habrá observado que la programación con archivos es muy sencilla. Usted puede determinar, leer o escribir cualquier registro determinado sin molestarse con los incómodos procedimientos necesarios para recuperar datos almacenados secuencialmente. Sin embargo, los métodos de inserción y eliminación que detallamos en la página 724 no constituyen el procedimiento más eficaz de tratar los archivos aleatorios; acceder a la información utilizando un índice es un método mejor.

Para crear un índice se debe especificar como clave un campo determinado del registro. Así pues, el índice construido se compone de los valores del campo de clave para cada registro, junto con su correspondiente número de registro. Por tanto, si el registro correspondiente a Juan Ruiz es el número 17, y es el octavo de una lista por orden alfabético, la matriz de índice almacenará un 17 en la octava posición. Si el índice está clasificado regularmente, buscar un registro puede ser un proceso muy rápido.

El índice se suele almacenar en RAM con el objeto de que sea accesible de forma inmediata. Se puede generar en el acto, con una rutina que lea todo el archivo, poniendo cada campo de clave en una matriz. Este índice se puede entonces clasificar para su utilización. Un método alternativo consiste en almacenar archivos de índices en disco al igual que archivos de datos. De esta forma, se pueden usar diversos campos como clave, mediante la creación en disco de diversos archivos índices de un mismo archivo. Esto también permitirá indexar el archivo de diferentes maneras: por ejemplo, los registros se podrían dar por orden de nombre (tanto de A a Z como de Z a A), por orden de fecha, etc.

¿Qué estructura debe tener un archivo índice? Cada registro de un archivo índice ha de contener dos campos (el dato de clave y el número de registro) para cada registro. Este registro se lee entero a la memoria, se utiliza y sólo se vuelve a escribir en disco si ha sido actualizado. Ésta es una aplicación ideal para un archivo secuencial en contraposición al archivo aleatorio, porque los datos se requieren en el orden en que están almacenados. Ésta es un área donde los archivos secuenciales y los archivos directos se complementan mutuamente.

Eliminar registros de archivos de acceso directo indexados es sólo cuestión de marcar los registros como eliminados y asegurarse de que ya no estén incluidos en el índice. La forma más económica de hacer esto consiste en grabar un indicador de "eliminado" en el registro (p. ej., un asterisco al principio del primer campo). La clave de ese registro se quita entonces del índice o, como alternativa, al número de registro se le puede poner algún valor especial que indique que el registro está eliminado (podría ser −1).

Cualquiera que sea el método elegido, lo importante es que en el archivo haya constancia de los registros eliminados. Cuando se añaden registros nuevos al archivo, éstos se pueden escribir sobre el espacio que ocupaba un registro eliminado. La entrada original del índice se debe reemplazar por una nueva y, en el momento apropiado, volver a clasificar el índice para que incluya al nuevo registro en la posición correcta dentro del archivo. De esta forma, el programa ofrecerá la posibilidad de recuperar registros eliminados accidentalmente, siempre y cuando en el ínterin no hayan sido sobreescritos.

Es una buena idea dar algún sistema para poner en orden el archivo índice. El sistema de indexación que hemos detallado es propenso a almacenar registros sin orden y con numerosos e innecesarios vacíos entre ellos. A medida que se trabaje con el archivo, la velocidad de acceso irá disminuyendo. La rutina de reorganización tendrá que clasificar los registros por el orden correspondiente y descartar del archivo todos los registros eliminados. La reorganización se puede llevar a cabo como una opción del usuario o de forma automática cada vez que el sistema concluye alguna operación de envergadura.

La indexación no es el único medio de acceder rápidamente a registros de un archivo grande. El *hashing* es un método alternativo muy apropiado para archivos muy grandes y, por consiguiente, se lo ve casi exclusivamente en sistemas de disco rígido o en máquinas con discos flexibles de gran capacidad. No obstante, muchos sistemas operativos y programas utilizan el hashing internamente para acelerar su operación; por tanto, se trata de una técnica que vale la pena conocer.

El hashing reemplaza al índice con una fórmula o algoritmo de hashing. Éste toma el valor del campo de clave y produce a partir de él un número de registro, al que se denomina *hash*. El registro correspondiente a la clave se almacena en esta posición del archivo. La fórmula será elaborada en función del tipo de datos del campo de clave. Si el campo de clave contiene una fecha (para posibilitar el almacenamiento cronológico de los registros), se podría emplear el número de mes, multiplicado por las dos últimas cifras del año, más el número del día. Un campo de nombre se podría "hashear" manipulando los códigos ASCII utilizados para las letras que componen el nombre, y así sucesivamente.

Supongamos que deseamos crear un archivo "hasheado" de registros de empleados utilizando como clave de clasificación los apellidos. El algoritmo de hashing que vamos a emplear es: tomar los códigos ASCII de las primeras cuatro letras y tratarlos como un número de ocho dígitos, elevar ese número al cuadrado, luego tomar los últimos cuatro dígitos del número resultante como el hash.



JONES, por consiguiente, se convierte en el registro 1161, mientras que JONQUIL se convierte en el 0161.

El hashing es muy distinto de un sistema indexado. Con el hashing sólo se puede tener un campo de clave (y un algoritmo de hashing) por archivo y éste se utiliza cuando se colocan por primera vez los registros en el archivo. Un archivo puede tener cualquier número de índices asociados, y éstos se pueden crear en cualquier momento, después o durante la creación del archivo.

El hashing es menos flexible que la indexación pero es mucho más rápido. Para hallar un registro determinado, el programa tan sólo toma la clave, le aplica el algoritmo de hash, y obtiene ese registro en particular. Por consiguiente, se ahorra el tiempo que lleva buscar en el índice, y también se gana el tiempo que cuesta crear el propio índice.

En el hashing se plantea un problema cuando dos registros generan el mismo código de hash y, por tanto, han de ocupar la misma posición en un archivo. Para evitar este problema, los algoritmos del hashing se diseñan cuidadosamente de modo que no haya dos claves (a menos que sean idénticas) que generen el mismo hash. Además, en el archivo los registros están espaciados de modo que dos hashes, que aparentemente están próximos entre sí, en realidad pueden tener entre ellos un espacio vacío de cinco registros.

Ahora podemos describir un sistema de hashing con mayor claridad de la siguiente manera. Cuando se almacena un registro, su clave se "hashea" para producir un número de registro. Si ese registro está ocupado, el sistema mira el siguiente registro en secuencia. Esto lo repite en los próximos cinco (o el número que sea) registros en busca de uno libre. Cuando se ha de recuperar un registro, se "hashea" su clave y entonces se busca secuencialmente en ese grupo de registros para hallar la clave buscada. Podría parecer que esto anula la ventaja de la velocidad, pero lo que el hashing hace efectivamente es reducir, quizá de tres mil a cinco o seis, el número de registros en los cuales buscar.

¿Qué sucede si todos los registros, los cinco o los que sea, para un hash determinado están ocupados? Hay varias formas de enfocar esto, siendo la más obvia de ellas emitir un mensaje de "archivo completo". Con mayor frecuencia, los registros que no se pueden colocar en su posición en el archivo se escriben en un archivo separado de *overflow* (exceso de capacidad) que posee su propio índice y se van incorporando al archivo principal cuando es posible. La mayoría de los sistemas se esfuerzan en evitar el *overflow*, manteniendo normalmente la ocupación de los archivos "hasheados" en un 80 % o menos de su capacidad. Esto evidencia otra limitación del acceso por hash a los archivos directos. Un archivo "hasheado" tiende a consumir más espacio que si se utilizara un índice.

El hashing también acelera la eliminación de registros no deseados. Simplemente se calcula el hash de la clave del registro, se efectúa una búsqueda rápida para localizarlo exactamente y se marca su posición como desocupada. Luego, la próxima vez que se agregue al archivo un registro con un hash idéntico, éste sobreescribirá al anterior.

En el capítulo final de esta serie analizaremos las instrucciones BASIC necesarias para crear archivos en cassette y acceder a ellos.

Unidos por índice

La forma más común de acceder a un archivo directo es mediante un índice. Éste es el esquema de una RAM que refleja los valores para un campo de clave determinado con los registros correspondientes. Cuando se está accediendo a un registro, se lo puede buscar rápidamente en el índice y leerlo luego a la memoria. Los registros eliminados permanecen en el archivo y se los marca como borrados. Luego se sobreescribe en ellos a medida que se van agregando registros nuevos

Encontrar "Davids"

Clave	N.°
Andreu	1
Benítez	−1
Bruch	5
Cruz	−1
Davids	7
Dávila	23
Ferrer	15
Gómez	28
Hernández	37
Jofré	25
Klaus	11
Martín	10

Archivo índice

Archivo principal

	Nombre	Tel. trab.	Tel. part.	Profesión
1	Andreu	242 07 91	727 09 42	Delineante
2	Pérez	636 24 18	221 39 40	Contable
3	Silva	631 08 36	286 81 70	Redactor
4		registro	eliminado	
5	Bruch	729 82 13	236 21 90	Dentista
6	Paredes	836 66 22	298 43 10	Fontanero
	Davids	743 72 16	450 69 26	Jardinero
8		registro	eliminado	
9		registro	eliminado	
10	Martín	730 63 21	429 75 92	Mecánico
11	Klaus	493 98 99	455 84 31	Abogado
12	Valverde	736 77 00	693 04 52	Peluquero

Haciendo un hash

Encontrar "Davids"

ALGORITMO DE HASHING

El algoritmo de hashing convierte las claves de modo que aludan a un determinado bloque de registros.
Los registros con hash idéntico se agrupan entre sí

Se deja espacio sin utilizar entre los bloques de registros, de modo que se puedan insertar registros nuevos en esas posiciones

Nombre	Tel. trab.	Tel. part.	Profesión
Davidson	629 04 91	430 05 92	Decorador
Daza	436 24 88	362 00 66	Sociólogo
Daroca	730 00 21	626 91 91	Tintorero
Damm	439 99 33	630 49 18	Escritor
Davids	743 72 16	450 69 26	Jardinero
Dávila	830 01 23	340 99 24	Programador
Esteruelas	731 66 66	458 00 21	Delineante
Eduardo	831 82 94	450 62 18	Proveedor

Los archivos "hasheados" ofrecen un acceso a gran velocidad a registros en grandes archivos aleatorios. No obstante, el sistema tiene sus limitaciones y exige una cuidadosa programación.
A la clave de registro se le hace corresponder una posición del archivo mediante la aplicación de un algoritmo de hashing preestablecido. Cada posible hash suele hacer referencia a un bloque de registros en el que se puede buscar secuencialmente para encontrar el registro deseado

Kevin Jones

Sonidos imaginativos

Pocos juegos hacen una utilización creativa del sonido. Veamos algunas ideas para desarrollar "audiojuegos" entretenidos

```
10 REM *** AUDIOJUEGO ***
20 PRINT "UN AUDIOJUEGO
SENCILLO": PRINT:PRINT "DIRIJA SU
NAVE HASTA EL RADIOFARO ANTES DE
QUE SE LE AGOTE EL COMBUSTIBLE"
30 PRINT:PRINT "CUANTO MAS
CERCA SE HALLE, MAS AGUDO SERA EL
TONO DEL RADIOFARO"
40 PRINT:PRINT "LOS MANDOS
SON:" PRINT "I (ARRIBA) M (ABAJO) J
(IZQUIERDA) K (DERECHA)"
60 PRINT:PRINT "***** PULSE UNA
TECLA PARA EMPEZAR *****"
70 A$=INKEY$(0):IF A$="" THEN
GOTO 70
80 P=INT(RND(1)*500)+1:Q=INT
(RND(1)*500+1):F=3*(P+Q)/2:X=0:Y=0
90 XD=0:YD=0
100 PRINT "COMBUSTIBLE=";F;
" DISTANCIA="
110 IF (ABS(P-X)<2 AND ABS
(Q-Y)<2) THEN PRINT "LO HAS
CONSEGUIDO CUANDO AUN TE
QUEDAN";F;"UNIDADES DE
COMBUSTIBLE":END
120 A$=INKEY$(0):IF A$="" THEN
GOTO 120
130 IF (A$="I" AND YD<3) THEN
YD=YD+1
140 IF (A$="M" AND YD>-3) THEN
YD=YD-1
150 IF (A$="J" AND XD>-3) THEN
XD=XD-1
160 IF (A$="K" AND XD<3) THEN
XD=XD+1
170 X=X+XD:Y=Y+YD
180 D=SQR ((P-X)*(P-X)+(Q-Y)
*(Q-Y))
190 PRINT D
200 SOUND 1,-8,(255-D),2
210 F=F-ABS(XD)-ABS(YD):IF F>
0 THEN GOTO 90
230 PRINT TAB(10);"***** CRASH
*****":PRINT TAB(11);"*** SE ACABO
EL COMBUSTIBLE ***"
240 PRINT:PRINT "LA POSICION
ES";D;"UNIDADES ESPACIALES
DESDE LA BASE"
260 END
```

Los juegos recreativos se valen del sonido para apoyar sus bien cuidados gráficos. Aunque los efectos sonoros se utilizan para aumentar el realismo y hacer más emocionante la acción, raramente constituyen el centro de atención del juego propiamente dicho. No obstante, si podemos utilizar nuestro sentido de la vista como la base para muchos y muy variados juegos por ordenador, no existe ninguna razón por la cual no podamos generar juegos que se centren en nuestro sentido del oído.

El que veremos aquí es uno de estos "audiojuegos". Aunque no pretendemos que se lo considere como el juego más emocionante jamás desarrollado, ciertamente es un juego muy interesante. Lo que al principio parece ser una tarea bastante sencilla enseguida se convierte en un desafío fascinante, sencillamente porque uno tiene que utilizar una entrada sensorial distinta de la que se suele emplear cuando se controla alguna visualización en pantalla con brillantes colores.

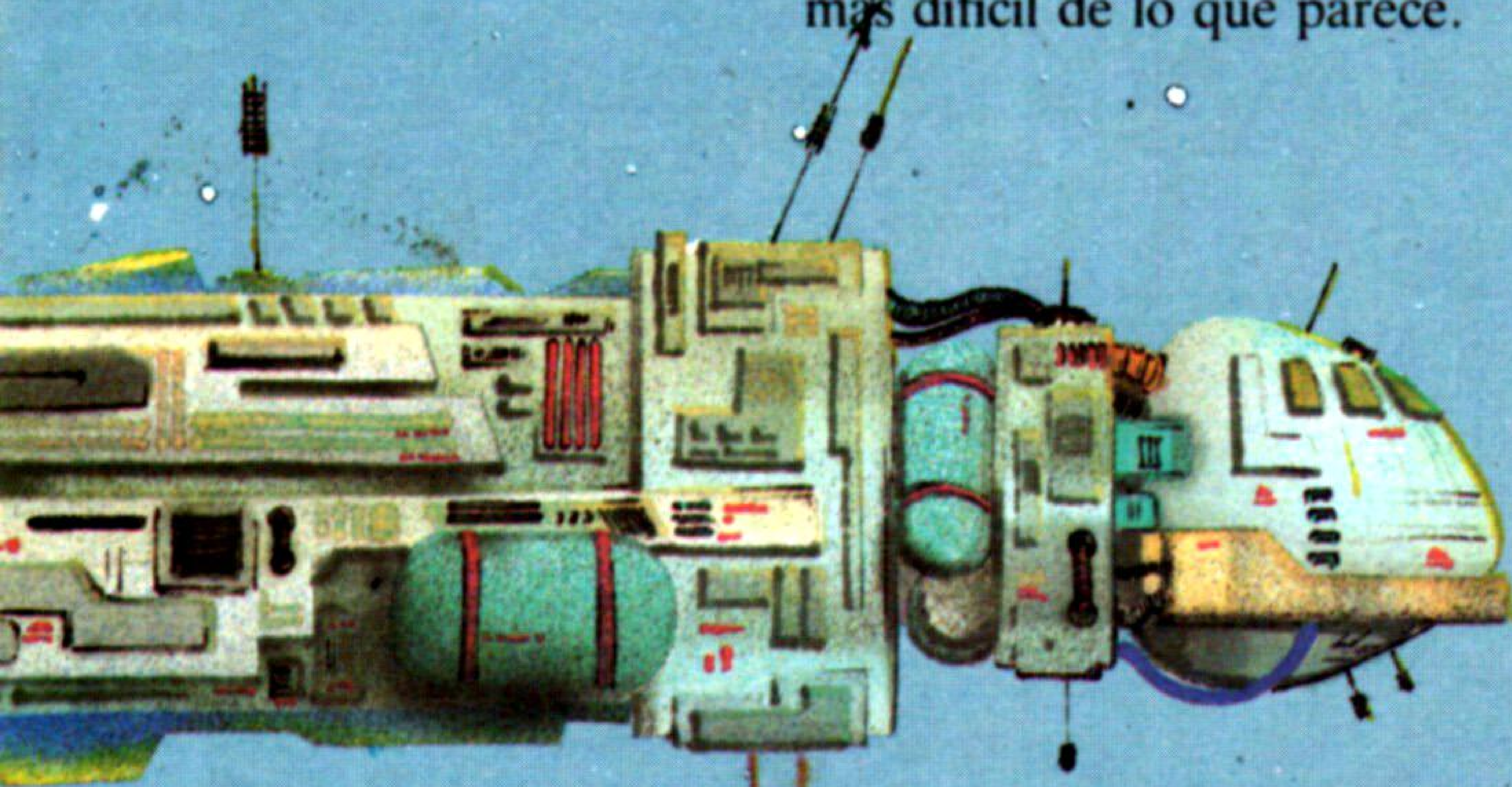
En nuestro juego usted está situado en los controles de una nave espacial que se está quedando sin combustible. Su única esperanza de supervivencia está en acoplarse a una estación de abastecimiento de combustible cercana. Lamentablemente, debido a un defecto de funcionamiento de su ordenador, carece de toda información visual para orientarse en sus esfuerzos para el acoplamiento; el único método de navegación es dirigirse hacia la estación utilizando una señal audible. El tono de la señal del "radiofaro de navegación" se va volviendo más agudo a medida que uno se va acercando; de modo que todo es cuestión de escuchar atentamente e ir respondiendo con precisión mediante los controles.

Al igual que en una nave espacial auténtica, una vez que se dirige la nave en una dirección determinada, seguirá yendo en esa dirección hasta contrarrestar ese movimiento utilizando el mando opuesto. Si utiliza dos U para avanzar rápidamente hacia arriba, entonces necesitará pulsar dos D para detenerse de nuevo. Esto hace que el juego sea mucho más difícil de lo que parece.

Se requerirá bastante práctica para ser capaz de completar el juego utilizando sólo las indicaciones de sonido. Sin embargo, agregando unas pocas líneas se puede dar una indicación en la pantalla acerca de dónde se halla la nave en relación a la estación de abastecimiento. Ésta podría sencillamente indicar la distancia entre la nave y la estación (la variable D) o podría informar al jugador acerca de cuáles son los botones más eficaces (si SGN(p−x)=−1, entonces hay que ir hacia la izquierda, p. ej.) Estas útiles pistas harán que el juego resulte mucho más sencillo.

Después de probar nuestro programa, quizá quiera crear sus propios juegos. Hacer uso del sonido para localizar o evitar objetos es un campo con considerables posibilidades de programación. ¿Qué le parecerían juegos de sonar para submarinos o un campo de minas en el cual fuera necesario navegar utilizando un detector que emite una señal audible?

Los efectos de sonido más avanzados de máquinas como el BBC Micro y el Oric-1 evidentemente serán una ventaja en este caso. Los juegos sonoros pueden utilizar varias voces de forma simultánea, o dotarlas de ruidos ligeramente diferentes, cada uno con su propio significado en el contexto del juego.



Dave Cooper-Smith

Complementos al BASIC

Este programa se escribió en un BBC Micro en Mode 7 y, por lo tanto, es BASIC Microsoft casi estándar. Sus instrucciones PRINT presuponen una visualización en pantalla de 40 columnas y habrá de reformatearse para tamaños distintos. La instrución SOUND de la línea 200 es exclusiva del BASIC BBC. El valor del parámetro (255-D) es (*pitch*) el de la nota a tocar, mientras que los otros parámetros controlan el volumen, la duración y el canal. INKEY$(0) en la línea 120, y el empleo de RND en la línea 80, requerirán algo de atención en otras máquinas:

Spectrum
Insertar LET en todas las sentencias de asignación. Cambiar INKEY$ (0) por INKEY$. Cambiar RND(1) por RND. Cambiar la línea 200 por:

```
200 BEEP 0.4,(255-D)
```

e insertar:

```
15 RANDOMIZE
195 IF D > 254 THEN LET D=D−254
```

Commodore 64 y Vic-20
Cambiar INKEY$(0) por GET A$. Remitirse al manual del usuario para las instrucciones de sonido del Commodore. Insertar:

```
75 X=RND (−TI)
```

Dragon
Cambiar INKEY$(0) por INKEY$. Cambiar RND(1) por RND(0). Cambiar la línea 200 por:

```
200 SOUND (255-D),10
```

e insertar:

```
195 IF D > 254 THEN D=D-254
```

Oric Atmos
Cambiar INKEY$(0) por KEY$. Cambiar la línea 200 por:

```
200 SOUND 1, (255-D), 9:WAIT 40:PLAY0,0,0,0
```

e insertar:

```
195 IF D>254 THEN D=D-254
```

Deporte acuático

"Scuba dive" constituye un refrescante cambio respecto a los juegos de disparos del tipo "invasores" y guerras intergalácticas

Existen a la venta tres versiones del *Scuba dive* (Submarinista): una para el Spectrum, otra para el Oric-1, y una tercera para el Commodore 64. La versión para el Oric-1 está siendo adaptada para que también funcione en el Atmos.

El jugador asume el papel de un submarinista que está recogiendo un tesoro del lecho marino, tarea en la cual arriesga su vida. El principal objetivo de nuestro intrépido héroe consiste en recoger perlas que acumulan puntos, que se extraen de ostras y de almejas gigantes. En una etapa más avanzada del juego se debe recoger un tesoro del interior de unos cofres que se hallan en las profundidades del cavernoso mundo submarino.

Sin embargo, existen muchas y buenas razones para ir avanzando con gran cautela. El agua está densamente poblada de criaturas que constituyen una mortal amenaza para el submarinista: medusas, pulpos, calamares, anguilas eléctricas y, en la versión para el Spectrum, ¡tiburones! Si le toca una de estas criaturas, se pierde una vida, aunque ninguna de ellas ataca expresamente; simplemente deben evitarse sea como sea. Otro peligro se descubre cuando se comienza a extraer perlas de las almejas gigantes: éstas tienen capacidad de cerrarse sobre uno, dejándolo atrapado.

Las estrechas entradas a la caverna principal y a las profundidades de nivel inferior están custodiadas por pulpos gigantes. Superar a los pulpos puede ser complicado, ya que sus tentáculos siempre están revoloteando por ahí. Pero, de vez en cuando, uno se las puede arreglar para deslizarse por entre ellos. En la versión para el Commodore no hay pulpos. En lugar de éstos hay una trampilla que impide el paso. Ésta se abre y cierra continuamente y hay que pasar por ella sin que un golpe fortuito lo deje inconsciente.

El programa concede tres vidas por juego y tiene cuatro niveles de dificultad. Se pierde una vida al tocar a alguno de los poco amistosos seres acuáticos o cuando se acaba el oxígeno.

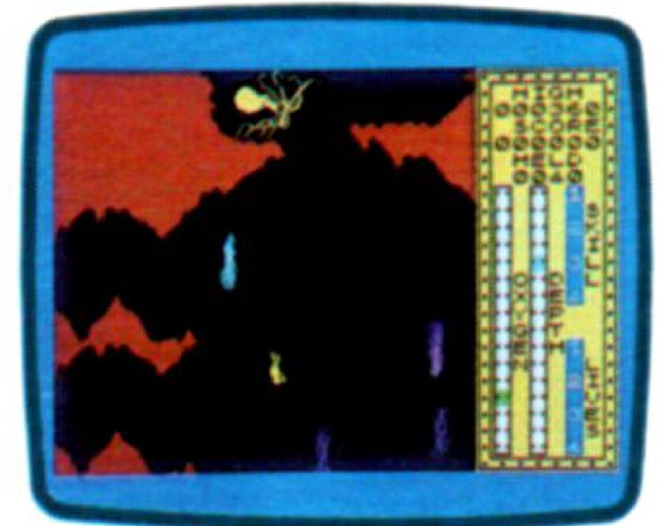
Cada versión del juego empieza con una panorámica de la superficie del mar y una gran parte de las profundidades. El barco desde el que uno se arroja está meciéndose en la superficie. Uno debe tener cuidado desde el mismo momento de tirarse al agua, ya que es posible quedar atrapado debajo de la embarcación. En la versión para el Spectrum es una ventaja tener un buen sentido de la orientación, porque el barco puede girar cuando uno se encuentra sumergido. En la versión para el Oric esto no representa un gran problema, ya que la embarcación siempre se mueve de izquierda a derecha sin que en ningún momento se salga de la pantalla y, en consecuencia, cuando uno emerge a la superficie siempre está a la vista.

La calidad de los gráficos en la versión para el Spectrum (que es con mucho la mejor de las tres) es soberbia. Se ha hecho buen uso del color y el diseño de las criaturas y de la caverna es de un gran realismo. El control sobre el submarinista se consigue utilizando las teclas X y Z para girar en el sentido de las agujas del reloj y en dirección contraria, respectivamente, y las teclas Space y Shift mueven al submarinista hacia adelante. Los usuarios del Spectrum también pueden emplear palancas de mando, aunque el programa no funciona con la interface para palanca de mando Kempston. Los usuarios del Commodore también disponen de la opción de la palanca de mando, pero quienes juegan con el Oric no disponen de este recurso y deben hacerlo con el teclado.

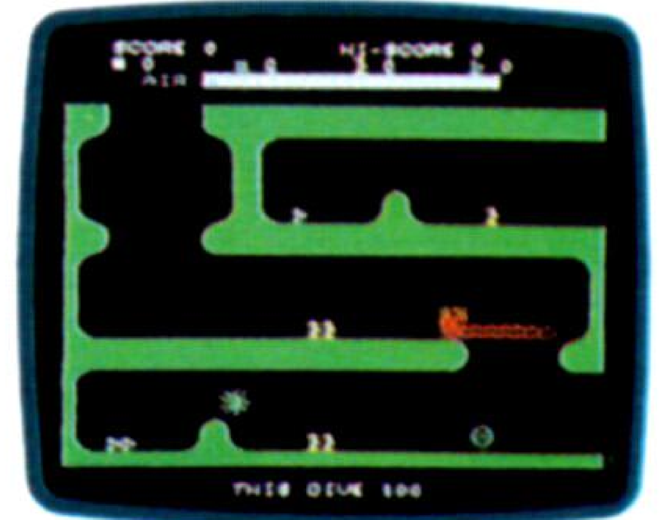
La versión para este último ordenador es, en varios sentidos, mucho menos emocionante que su equivalente para el Spectrum. El movimiento del submarinista y de los seres marinos es muy torpe, y los gráficos (en especial las paredes de las cavernas y los cofres) están mucho menos detallados.

Por su parte, la versión para el Commodore tampoco alcanza el nivel de calidad de imagen que caracteriza a la del Spectrum, pero, sin lugar a dudas, es mucho más satisfactoria que la del Oric.

Comparando calidad
Estas fotografías ilustran la diferencia de calidad entre las distintas versiones del juego *Scuba dive*



Sinclair Spectrum



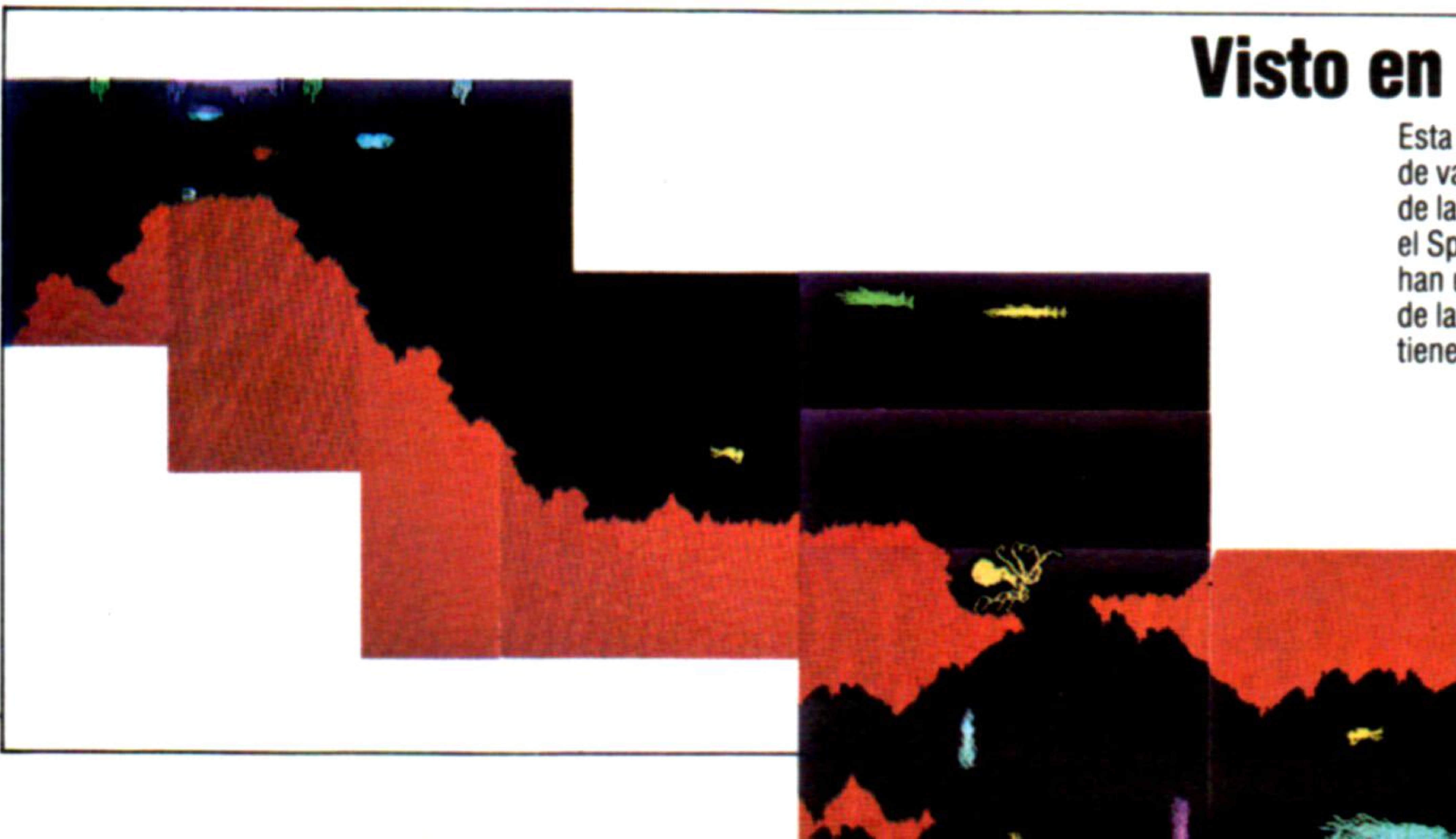
Oric-1



Commodore 64

Visto en perspectiva

Esta imagen se compuso a partir de varios volcados de pantalla de la versión del *Scuba dive* para el Spectrum. Las imágenes se han unido para mostrar la visión de las cavernas submarinas que tiene el submarinista



Ian McKinnell

Restar como sumar

Ha llegado el momento de estudiar las operaciones aritméticas de suma y resta en código máquina

Precisamente son las operaciones aritméticas las que nos van a mostrar las filosofías de diseño que diferencian al Z80 del 6502. Los numerosos registros del Z80, con su sofisticado juego de instrucciones, dan idea del procesador mismo: elegante, complejo y potente. En cambio, la arquitectura y el conjunto de instrucciones del 6502, mucho más sencillas, indican un procesador más modesto, robusto y práctico sin duda, pero carente, en apariencia, de la categoría de un Z80. A pesar de todo, la riqueza de tipos de direccionamiento del 6502 y el empleo que hace de la página cero como un registro índice adicional le otorgan su carácter ágil y versátil, y sugieren que su actual hegemonía dentro del mundo de los microordenadores personales y de oficina va a durar todavía tiempo.

La ventaja que tienen los registros del Z80 es que son flexibles. Simultáneamente pueden ser tratados como registros de un solo byte o de dos, permitiendo un gran ámbito de direccionamiento. Por el contrario, los registros de dos bytes están ausentes en el 6502, aunque gracias a sus tipos de direccionamiento puede servirse de la página cero como si fuera una tabla de registros de uno y de dos bytes.

Aritmética básica

Ya hemos visto que los registros de la CPU permiten una gran variedad en los posibles accesos a memoria, pero ahora veremos que el manejo de la memoria comporta algo más que cargas, descargas y comparaciones de su contenido. Para un sistema, la posibilidad de realizar las cuatro operaciones básicas de la aritmética es fundamental, y es curioso que tanto el Z80 como el 6502 tan sólo sepan sumar y restar. La multiplicación y la división, e incluso la suma y resta de números superiores a $FF, deben ser previamente programadas. Esto constituye una limitación evidente de ambas CPU, pero su valor pedagógico es incalculable, ya que el programador ha de inventarse los algoritmos para multiplicar y dividir. Los procesadores de 16 bits que han aparecido después del Z80 y del 6502 pueden realizar también estas dos operaciones, debido a la mayor potencia y rapidez de sus CPU.

Para el tratamiento aritmético de un solo byte a la vez ya hemos tenido ocasión de utilizar la instrucción ADC (*ADd with Carry:* sumar con un bit de arrastre) y varias otras del tipo INC (INCrementar) en ambas CPU. En el cuadro adjunto tenemos un ejemplo de cómo sumábamos los contenidos de dos posiciones de memoria cada una de ellas con dos bytes numéricos. El 6502 recurre al método de byte a byte que el Z80 conoce, pero a éste le resulta más fácil emplear otro método basado en sus pares de registros, de los que no hay equivalentes en el 6502. Nótense las estrategias que emplean ambos procesadores para las diversas posibilidades de arrastre en la suma: se comienza con instrucciones como

6502	Z80
ADDR1 DW $7E60	ADDR1 DW $7E60
ADDR2 DW $4A51	ADDR2 DW $4A51
SUM DS $03	SUM DS $03
BEGIN CLC	BEGIN LD A,$00
LDA ADDR1	AND A
ADC ADDR2	LD HL,(ADDR1)
STA SUM	LD DE,(ADDR2)
LDA ADDR1+1	ADD HL,DE
ADC ADDR2+1	LD (SUM),HL
STA SUM+1	ADC A,$00
LDA $00	LD (SUM+2),A
ADC $00	RET
STA SUM+2	
RTS	

CLC (6502) o bien AND A (Z80) que borran, antes de hacer la suma, el flag de arrastre poniéndolo a cero, y se acaba modificando el tercer byte de SUM, para el caso de que el resultado desborde los dos bytes, caso que jamás se ha de descuidar.

De la misma manera que tratamos la suma se puede hacer con la resta, ya que ambos procesadores poseen la instrucción SBC (*SuBtract with Carry*) aunque el Z80 puede hasta restar dos bytes a la vez. Pero como la resta nos introduce en el tema de los números negativos, hemos de hacer una pequeña incursión en la representación binaria del signo algebraico.

Para empezar, de los números negativos está dicho todo con entender bien esta expresión:

Si $A + B = 0$ entonces $A = -B$

Diremos en este caso que B es el negativo o complemento de A, ya que sumados dan cero. Si tenemos en cuenta los números que caben en un solo byte, no es sorprendente considerar el número $FC como el complemento de $04, ya que

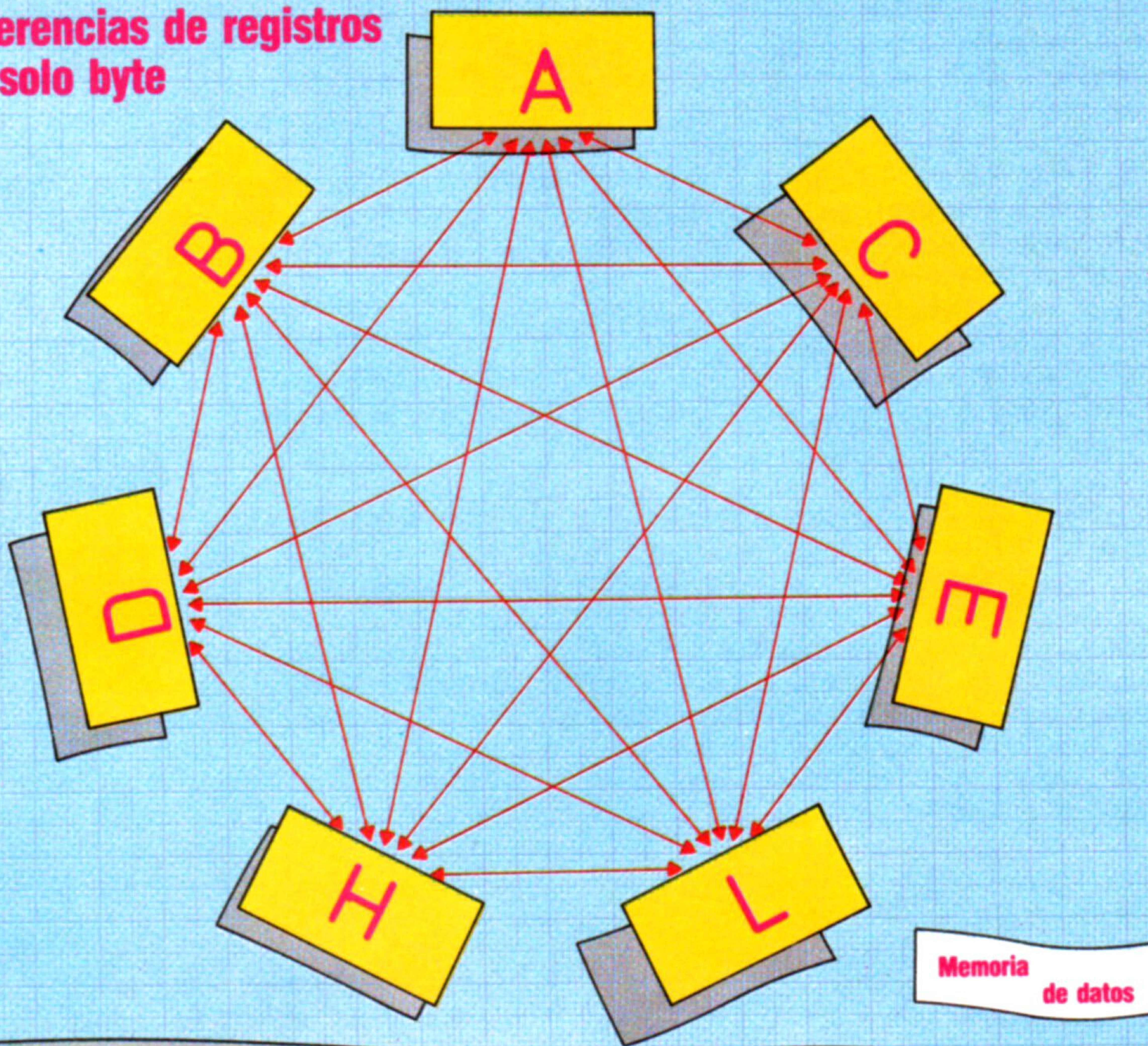
$04 + $FC = $00

Note que el resultado en realidad sería $100 pero ya advertimos que sólo se dispone de un registro con capacidad para un solo byte. Así pues, ambos números son complementarios, o el uno negativo del otro. Este pequeño descubrimiento nos conduce a una conclusión algo chocante pero utilísima: restar es sumar dos números, sustituyendo uno de ellos por su negativo. Es decir:

La resta $A - B$ equivale a la suma $A + (-B)$

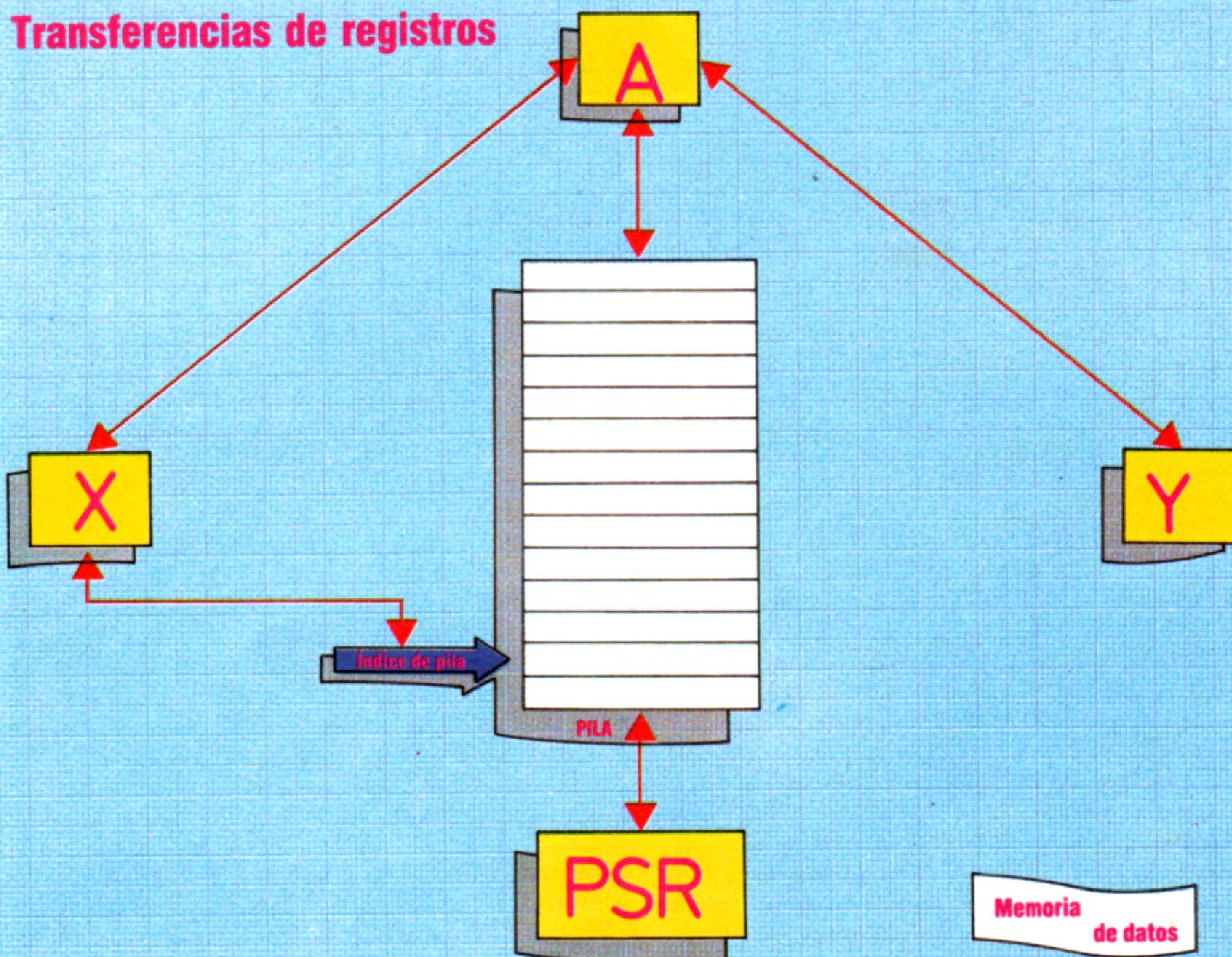
Si seguimos teniendo en cuenta el registro de un solo byte, la siguiente resta $09 − $04 se convierte en la suma $09 + $FC, pues ya vimos que el complemento de $04 es $FC. El resultado es $05, que es lo que cabe en un byte, aunque en realidad debería ser, para la suma, $105.





Kevin Jones

Doble personalidad
Los registros de datos del Z80 pueden comportarse como registros de un byte y comunicar con cualquier otro registro también de un solo byte. Cualquiera de ellos puede comunicar con la memoria en el modo de direccionamiento directo, inmediato, indirecto, absoluto e indexado. Si son tratados en pareja, como BC, DE, HL, pueden transferir datos de 16 bits de la memoria a la pila y viceversa, y funcionar como acumuladores de 16 bits en sumas y restas. Esta mezcla de flexibilidad y riqueza de recursos es lo que explica el inmenso éxito del Z80



Kevin Jones

Claro y sencillo
La comunicación interna del 6502 es estrictamente lineal y limitada a transferencias de datos de 8 bits. Solamente el acumulador puede comunicarse directamente con el X y el Y. Tan sólo X puede comunicarse con el índice de pila; así como los únicos que tienen acceso a la pila son el indicador de estado y el acumulador. Las transferencias de la memoria son posibles en los modos absoluto, directo, indirecto, indexado, inmediato y página cero. El original empleo que hace el 6502 de la página cero compensa el reducido tamaño de sus registros, pues la página cero puede ser tratada como registros de la CPU de dos bytes. ¡Nada menos que 128 registros disponibles!

Aquí es esencial la pregunta ¿quién es negativo de quién? Ya que está claro que si $FC es el negativo de $04, entonces $04 es el negativo de $FC. Es norma convenida, por una razón que vamos a explicar, considerar POSITIVOS los números (¡siempre de un solo byte!) que van desde el $00 al $7F, mientras serán NEGATIVOS los comprendidos entre $80 y $FF. En decimal, de los 256 números posibles en un byte, la mitad inferior de ellos son de signo positivo, y la otra mitad, negativo. Y la razón de esto radica en que los números inferiores a 128, representados en binario sobre un byte, tienen el bit superior (el bit 7) siempre a cero (p. ej., en binario 127 es 0111 1111); mientras que el bit 7 de los binarios comprendidos entre 128 y 255, ambos inclusive, está siempre a uno (p. ej., 255 es 1111 1111; 128 es 1000 0000). Era, pues, lógico que se acordara que el bit 7 fuera el *bit del signo*, y que el flag de arrastre se activara o desactivara según indicara el bit 7 del byte de resultado en una operación aritmética o lógica. Con esto se abre la posibilidad de que el resultado sea negativo.

Otro concepto por explicar es el del *complemento a 1*. Hasta aquí hemos utilizado, sin darnos cuenta, la noción de *complemento a 2*: el complemento a 2 de $04 es $FC, y al revés, pues sumados dan $00 en un solo byte. Pues bien, el complemento a 1 de un número expresado en binario como

0000 1101 (hexa 0D)

se obtiene fácilmente poniendo cada uno de sus bits en su estado opuesto, cambiando los ceros por unos, y los unos por ceros:

1111 0010 (hexa F2)

¿Qué utilidad le reporta saber esto? La siguiente: que basta con añadir un 1 al complemento a 1 para convertirlo en complemento a 2. En nuestro ejemplo, el complemento a 2 del número binario de partida será 1111 0011 (hexa F2 + 1). Note finalmente que en hexadecimal un número y su complemento a 1 suman siempre $FF (en el ejemplo, 0D + F2 = FF), y que este resultado se convierte en $100 con sólo sumarle 1. Bien entendida esta lección, usted no tendrá dificultad alguna con las explicaciones del próximo capítulo. Capítulo en el cual anticipamos que trataremos, además, de los algoritmos de multiplicar y dividir.

Respuestas a los ejercicios de p. 739

1) Éste es el programa que permite invertir el orden del string contenido en LABL1:

6502

```
;
ORIGIN   ORG   $7000
LAST1    EQU   $0D
LABL1    DB    'THIS IS A MESSAGE'
TERMN8   DB    LAST1
;
BEGIN    LDX   #$FF
         LDA   #LAST1
         PHA
LOOP0    INX
         LDA   LABL1,X
         PHA
         CMP   #LAST1
ENDLP0   BNE   LOOP0
CLRSTK   PLA
;
BEGIN1   LDX   #$FF
LOOP1    INX
         PLA
         STA   LABL1,X
         CMP   #LAST1
ENDLP1   BNE   LOOP1
         RTS
```

En la versión para el 6502, las instrucciones que van desde LOOP0 hasta ENDLOOP0 emplean el direccionamiento indexado con X en un bucle que sirve para cargar uno a uno los caracteres de LABL1 y "empujarlos" (*push*) en la pila, colocando en la pila primero el valor ASCII del carácter (') para indicar el fondo de la pila. Este mismo valor será también el último que se "empuje" en la pila, indicando ahí el final del string. Con él concluimos el bucle, y una vez conseguido esto lo borramos de la pila en CLRSTK (*clear stack*: limpiar pila).

En el caso del Z80, se usa IX en modo de direccionamiento indirecto para cargar desde LABL en adelante, y coloca en la pila no sólo el acumulador sino hasta el registro flag. Esto quiere decir que en la pila aparecerán intercalados los caracteres del string con los sucesivos valores del indicador de estado.

Z80

```
         ORG   $C000
LAST1    EQU   $0D
LABL1    DB    'THIS IS A MESSAGE'
TERMN8   DB    LAST1
;
BEGIN    LD    IX,LABL1-1
         LD    A,LAST1
         PUSH  AF
LOOP0    INC   IX
         LD    A,(IX+0)
         PUSH  AF
         CP    LAST1
ENDLP0   JR    NZ,LOOP0
CLRSTK   POP   AF
;
BEGIN1   LD    IX,LABL1-1
LOOP1    INC   IX
         POP   AF
         LD    (IX+0) ,A
         CP    LAST1
ENDLP1   JR    NZ, LOOP1
         RET
```

En ambos casos, la codificación entre BEGIN1 y ENDLP1 es similar a la del bucle anterior, pero extrayendo (POP,PLA) de la pila los caracteres en orden inverso, los últimos al principio, y almacenándolos en LABL1. El bucle concluye al encontrar el carácter que dejamos como señal al fondo de la pila.

Hay que remarcar la importancia de administrar bien tanto los llenados de la pila como sus vaciados (PHA o PUSH, PLA o POP), y también es importante saber manejar las condiciones de los extremos (qué se hace para iniciar el bucle, para acabar y para darle continuidad).

La instrucción LD IX,LABL1−1 que encontra-

mos dos veces en la rutina para el Z80 es un ejemplo de la utilidad de un programa ensamblador. Aquí decodifica la expresión LABL1−1 como diciendo "la dirección del byte que antecede inmediatamente al que tiene por dirección LABL1", y ensambla esa dirección en el código generado. La mayoría de los ensambladores admite cierta medida de evaluación de expresiones, permitiendo también normalmente la suma o resta de alguna constante a los operandos.

2) Este programa invierte el orden de los caracteres en cada palabra del string contenido en LABL1, conservando el orden de estas mismas palabras:

6502

```
;
ORIGIN    ORG    $7000
LAST1     EQU    $0D
SPACE     EQU    $20
LABL1     DB     'THIS IS'
TERMNB    DB     LAST1
;
BEGIN     LDX    #$FF
LOOP0     JSR    RVSWRD
          CMP    #LAST1
ENDLP0    BNE    LOOP0
          RTS
;
;****REVERSE A WORD S/R****
LASTCH    DB     $00
LASTX     DB     $00
RVSWRD    TXA
          TAY
          INY
RVSLP0    INX
          LDA    LABL1,X
          PHA
          CMP    #SPACE
          BEQ    CLRSTK
          CMP    #LAST1
ENDRV0    BNE    RVSLP0
CLRSTK    PLA
          STA    LASTCH
          STX    LASTX
RVSLP1    PLA
          STA    LABL1,Y
          INY
          CPY    LASTX
ENDLP1    BNE    RVSLP1
          LDA    LASTCH
          RTS
```

Varios son los detalles que nos interesan aquí (el empleo de JSR y CALL, p. ej.). La subrutina RVSWRD (invertir palabra) se parece a la que encontramos en el ejercicio 1, pero invierte sólo los caracteres dentro de cada palabra. En ambas versiones, la del 6502 y la del Z80, se usa el registro índice (el X y el IX respectivamente) para pasar a la subrutina la dirección de inicio de la palabra, y el acumulador es utilizado para devolver al programa principal el valor del carácter de fin de tarea (que puede ser tanto un espacio como la comilla). Se trata de una técnica de paso de valores muy común en lenguaje assembly, pero se ha de emplear con cuidado, sobre todo si se tiene la costumbre de llevar a la pila todos los registros de la CPU al comienzo de cada subrutina (como se dijo en la p. 738).

Otro punto relevante es el empleo del registro Y en la versión del 6502, utilizado primero para guardar la dirección de inicio de la palabra (mientras el X servía de índice en el bucle que llena la pila), y después sirviendo a su vez de índice en el bucle de vaciado de ésta (mientras el X guarda la dirección del final de la palabra). Decimos "dirección" con imprecisión, pues tanto X como Y son registros de un solo byte, incapaces de contener una dirección completa (que se compone de dos bytes, como sabemos). De todos modos, aquí lo que contienen es el byte de desplazamiento respecto a la dirección de LABL1. El Z80 no tiene problemas: tanto IX como IY pueden contener una dirección completa.

Z80

```
          ORG    $C000
LAST1     EQU    $0D
SPACE     EQU    $20
LABL1     DB     'THIS IS A MESSAGE'
TERMN8    DB     LAST1
;
BEGIN     LD     DE,LABL1-1
LOOP0     CALL   RVSWRD
          CP     LAST1
ENDLP0    JR     NZ,LOOP0
          RET
;
;***REVERSE A WORD S/R***
LASTCH    DB     $00
RVSWRD    PUSH   DE
          POP    HL
          INC    HL
RVSLP0    INC    DE
          LD     A,(DE)
          PUSH   AF
          CP     SPACE
          JR     Z,CLRSTK
          CP     LAST1
ENDRV0    JR     NZ,RVSLP0
CLRSTK    POP    AF
          LD     (LASTCH),A
;
RVSLP1    POP    AF
          LD     (HL),A
          INC    HL
          LD     A,L
          CP     E
          JR     NZ,RVSLP1
          LD     A,H
          CP     D
ENDRV1    JR     NZ,RVSLP1
          LD     A,(LASTCH)
          RET
```

No se utilizan IX ni IY en la versión del Z80, pero sí el par de registros HL y DE. Al igual que X e Y en el 6502, sirven para guardar el comienzo y final de las direcciones de la palabra, pero en lugar de servir de índice a una dirección de base, funcionan como direcciones indirectas: la instrucción LD A,(DE) quiere decir "carga el acumulador con el contenido del byte cuya dirección está indicada en DE". Todos los pares de registros del Z80 se pueden usar así. Una limitación es la falta de una instrucción que compare dos bytes; para comparar DE y HL hay que comparar E con L y después D con H. Igualmente, en la versión 6502, se comparan X e Y de modo indirecto, por medio de una posición de memoria.

El secreto del éxito

Puede considerarse excepcional el hecho de que en la lista de los diez programas más vendidos no aparezca alguno de los creados por la empresa británica Quicksilva

La fabricación del microordenador Sinclair ZX80 fue el impulso inicial para Quicksilva. El éxito de la máquina alentó a un ingeniero de pruebas, Nick Lambert, trabajador por cuenta propia, a diseñar una placa accesoria de tres Kbytes para complementar la escasa memoria de un Kbyte del ZX80. Lambert obtuvo un gran éxito vendiendo esta placa por correo. Cuando al año siguiente Sinclair lanzó el ZX81, Lambert creó Quicksilva, junto con John Hollis y Mark Eyles, para producir una serie de accesorios para la nueva máquina. En 1981 también apareció *Defender*, un juego escrito por Lambert, que fue la primera incursión de Quicksilva en el mercado del software.

El éxito de su juego *Defender* alentó a Quicksilva a producir más software de tipo recreativo, y finalmente se abandonó la vertiente de hardware de la empresa. En abril de 1982 se lanzó Quicksilva como una sociedad anónima.

El segundo gran éxito de software de la empresa fue un juego de aventuras llamado *Timegate*. En la Navidad de 1982 ya había 10 juegos Quicksilva en el mercado, y en enero del año siguiente W.H. Smith encargó 10 000 ejemplares de *Timegate*. Con sus productos (y su nombre) penetrando en el mercado de software comercial, que estaba en rápida expansión, la demanda de productos de Quicksilva subió como la espuma. Habiéndose lanzado con un descubierto de 200 libras, el movimiento total de la empresa durante su primer año de operación fue de 70 000 libras. Los productos Quicksilva ahora se suministran a través de grandes cadenas de tiendas y 150 detallistas independientes. La empresa cree que sus juegos cubren más del 70 % del mercado mundial en la actualidad.

El repentino crecimiento de la demanda significó que la empresa tuviera que ampliarse rápidamente, superando el funcionamiento basado en tres hombres. En la actualidad Quicksilva publica anuncios en la prensa especializada pidiendo autores de software, y un escritor de juegos puede percibir hasta el 15 % por cada una de las cassettes de sus programas que se vendan, en concepto de royalties. Hoy la empresa continúa diversificándose y ya ha dejado de crear exclusivamente para las máquinas Sinclair: el catálogo actual de Quicksilva figura en la actualidad entre los más variados del mercado, por cuanto incluye juegos diseñados para el BBC Micro, el Dragon, el Commodore 64 y el Vic-20.

Quicksilva está considerada en estos momentos como una de las principales empresas editoriales de software de Gran Bretaña. Su prestigio se acrecentó aún más debido al enorme éxito de *Ant attack* (véase p. 486), juego que realizaba gráficos sumamente sofisticados.

Rod Cousens, que asumió el cargo de director gerente de la empresa cuando Lambert decidió concentrarse en proyectos más creativos, fue elegido vicepresidente del Gremio de Casas de Software y "Persona del año" por la Computer Trade Association británica el año 1983.

Quicksilva continúa buscando caminos inesperados hacia los cuales diversificarse. En particular, la empresa ha lanzado recientemente un juego "no violento" denominado *The snowman* (El muñeco de nieve), que está basado en un cuento para niños de Raymond Briggs. Se lo considera como un antídoto, que ha sido bien recibido por el público, contra la enorme plétora de juegos "de disparos" del tipo "invasores del espacio".

En otros sentidos, los planes de la empresa son un poco más predecibles: espera producir enseguida un gran impacto en el mercado de software norteamericano.

Mapa cortesía de Sinclair

A la venta
Éstos son algunos de los últimos juegos que Quicksilva ha sacado a la venta. En la actualidad la empresa produce juegos para una amplia gama de ordenadores



Ian McKinnell



Rod Cousens
Actual director gerente de Quicksilva



Mark Eyles
Director de publicidad y uno de los fundadores de la empresa

Cortesía de Quicksilva


Editorial Delta, S.A.
mi computer



10038

9 788485 822836